ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DE LA SALUD

REGISTRO REMOTO DE DATOS BIOMÉDICOS "ONLINE"

ON-LINE REMOTE REGISTRY OF BIOMEDICAL DATA

Realizado por
Iván José Alba García
Tutorizado por
Manuel Jesús Martín Vázquez
Rafael Jesús Navas González
Departamento
Electrónica

UNIVERSIDAD DE MÁLAGA
MÁLAGA, junio 2016

Fecha defensa:
El Secretario del Tribunal

Resumen: Este trabajo fin de grado consiste en diseñar e implementar un sistema básico de bajo coste para el registro remoto de datos biomédicos de pacientes in situ. Este sistema está formado por un sistema de adquisición de datos compuesto por la placa Arduino UNO y varios módulos adicionales, como el módulo WiFi para la conexión a la red y la plataforma eHealth con sus distintos sensores para captar las variables fisiológicas del cuerpo humano. Además también se compone de un sistema de recepción y gestión de los datos ubicados en un servidor web, compuesto por una base de datos, un archivo de recepción de datos y una aplicación web.
La iniciativa en la comunicación y transmisión de la información la lleva a cabo siempre el sistema de monitorización, ya que cuando se desee obtener un registro, el sistema de adquisición tomará la muestra y la enviará mediante comunicación WiFi al sistema de recepción y gestión. Es allí donde ésta se confirmará para su posterior almacenamiento en la base de datos.
Desde la aplicación web diseñada se podrá acceder con distintos perfiles de usuarios para gestionar y visualizar mediante gráficas o numéricamente toda la información del sistema, gracias a la realización de consultas a la base de datos.

Palabras claves: Monitorización de variables fisiológicas, Sensores biomédicos, Arduino y e-Salud, Red inalámbrica, Servidor web

Abstract: This degree work consists of designing and introducing a basic low cost system for the remote registration of biomedical patient data in situ. This system is formed by a data acquisition system composed of Arduino UNO board and many additional pieces, like WiFi for network connection and eHealth with its different sensors that capture physiological variables in the human body. In addition, it is formed by a reception and management data located on a web server, formed by a database, a file data reception and a website.
The initiative on the communication and transmission of information is made by the monitoring system, because when it is needed to get a register, the acquisition system will take the sample and will send it by WiFi to the reception and management system. It is there where the information sent will be confirmed for its subsequent storage in the database.
From the web application designed will be possible to access with different user's profiles to manage and visualise with graphics or numerically all system's information, thanks to database search.

Keywords: Monitoring of physiological variables, Biomedical Sensors, Arduino and eHealth, Wireless network, Web Server.

Tabla de contenido

Índice de figuras

Índice de tablas

Glosario

AES: Advanced Encryption Standard.

AHA: American Heart Association (Asociación Estadounidense del Corazón).

ARP: Address Resolution Protocol (protocolo de resolución de direcciones).

ASCII: American Standard Code for Information Interchange (Código Estándar Estadounidense para el Intercambio de Información).

CPU: Central processing unit (Unidad de procesamiento central).

DHCP: Dynamic Host Configuration Protocol (Protocolo de configuración dinámica de host).

DNS: Domain Name System (Sistema de Nombres de Dominio).

EEPROM: Electrically Erasable Programmable Read-Only Memory (ROM programable y borrable eléctricamente).

FTP: File Transfer Protocol (Protocolo de Transferencia de Archivos).

GLCD: Graphic Liquid Crystal Display (Pantalla gráfica de cristal líquido).

GPIO: General Purpose Input/Output (Entrada/Salida de Propósito General).

GPL: General Public License (Licencia Pública General).

HTML: HyperText Markup Language (Lenguaje de marcas de hipertexto).

HTTP: Hypertext Transfer Protocol (Protocolo de transferencia de hipertexto).

ICMP: Internet Control Message Protocol (Protocolo de Mensajes de Control de Internet).

IDE: Integrated Development Environment (Entorno de desarrollo integrado).

JSON: JavaScript Object Notation (Notación de Objetos de JavaScript).

PHP: Hypertext Preprocessor.

PSK: Pre-Shared Key (Clave precompartida).

Rx: Receiver (Recepción).

SGDB: Sistema gestor de base de datos.

SHA1: Secure Hash Algorithm (Algoritmo de Hash Seguro).

SQL: Structured Query Language (Lenguaje de Consulta Estructurado).

SRAM: Static Random Access Memory (Memoria estática de acceso aleatorio).

SSL: Secure Sockets Layer (Capa de puertos seguros).

TCL: Tool Command Language (lenguaje de herramientas de comando).

TCP: Transmission Control Protocol (Protocolo de Control de Transmisión).

TKIP: Temporal Key Integrity Protocol.

TLS: Transport Layer Security (Seguridad de la capa de transporte).

Tx: Transmission (Transmisión).

UART: Universal Asynchronous Receiver-Transmitter (Transmisor-Receptor Asíncrono Universal).

UDP: User Datagram Protocol (Protocolo de datagrama de usuario).

UTF: Unicode transformation format (Formato de transformación Unicode).

WPA: WiFi Protected Access (Acceso WiFi protegido).

Capítulo 1. Introducción

1.1. Motivaciones

La tecnología y la informática cada vez están más presentes en nuestras vidas, basta con mirar hacia cualquier lado para ver algún que otro dispositivo electrónico cerca de nosotros. Como no podría ser de otro modo, el campo de la salud también avanza en este sentido y a pasos agigantados. Hoy en día casi toda la gestión sanitaria está informatizada, ejemplos de ello son los historiales clínicos digitales y las recetas electrónicas, que hace que todo sea más ordenado y eficiente.

Este proyecto se basa en unir distintas tecnologías como son: Sitio web, comunicaciones inalámbricas, base de datos y sistemas basados en microcontrolador, para conseguir un sistema completo de registro, transmisión, almacenamiento y gestión de la información.

Así, las tareas de este trabajo comprenden el estudio de las necesidades y requerimientos del sistema, la elección de los elementos hardware, y el desarrollo de las aplicaciones software apropiadas para su implementación. Estas tareas cubren aspectos relativos a la concepción e implantación de un sistema básico de e-Salud, por lo que resulta ser un proyecto completo y adecuado para un trabajo fin de grado de un Graduado en Ingeniería de la Salud.

Por otro lado otra de las grandes motivaciones es la de ayudar a la sociedad, poniendo a la disposición de los que lo necesiten un servicio con un bajo coste que le permita estar controlados y vigilados por un médico, y de esta forma poder diagnosticar e intervenir rápidamente. Un sistema como este ayudaría a evitar traslados innecesarios al hospital para una simple revisión de medidas fisiológicas básicas ya que estas medidas se podrían realizar desde casa.

1.2. Objetivos

El principal objetivo de este trabajo fin de grado se centra en realizar un sistema de registro de datos biomédicos donde el que inicia la comunicación e intercambio de datos es el mismo sistema de monitorización.

La segunda parte de este TFG se centra en desarrollar un sistema de recepción y gestión para almacenar y mostrar los datos recibidos del sistema de monitorización, donde se incluyen el diseño de una base de datos y una aplicación web.

Por lo tanto, los objetivos específicos a desarrollar en este trabajo son:

- Estudio, utilización e integración de diferentes componentes tanto hardware como software para el desarrollo de una aplicación e-Salud.
- Especificación, diseño e implementación de un sistema para la adquisición y el registro de datos biomédicos usando un dispositivo formado por una placa Arduino junto con la plataforma eHealth.

- Establecer un intercambio de información desde el microcontrolador Arduino hasta el Servidor Web mediante conexión WiFi, donde la iniciativa en la comunicación y transmisión de la información la lleva el sistema de adquisición de datos.
- Creación y actualización de una base de datos, con el fin de almacenar la información requerida.
- Comprobación y validación de los datos recibidos por parte del sistema de recepción y gestión de datos.
- Creación de una interfaz de usuario, donde se puedan visualizar toda la información almacenada en la base de datos.

1.3. Estructura del documento

Este documento está estructurado en base a 3 capítulos principales en los que se abordan los distintos aspectos contemplados para la ejecución de este proyecto.

En el capítulo 1 se encuentra la introducción del proyecto, poniendo en manifiesto las principales motivaciones y objetivos que se han tenido en cuenta para la realización de este trabajo fin de grado.

El capítulo 2 está dedicado a la exposición de la estructura del sistema diseñado, así como a presentar cada una de las partes que compone este sistema.

En el capítulo 3 se lleva a cabo la explicación de la implementación del sistema, tanto el sistema de adquisición de datos como el de recepción y gestión de datos.

En el apartado de Conclusiones y líneas futuras se resumen los principales logros de este TFG, así como las posibles líneas de desarrollo.

En el apartado Bibliografía, se hace una relación de la bibliografía consultada para la realización del proyecto.

Por último, en el aparto Apéndices, se recoge el material extra, como el presupuesto de construcción del prototipo y los códigos de programación desarrollados para la realización del sistema.

Capítulo 2. Arquitectura del sistema

La idea general del proyecto es diseñar un sistema que registre variables fisiológicas de los pacientes utilizando componentes de bajo coste. Para ello se ha utilizado Arduino UNO junto con la plataforma eHealth, que incluye una placa y varios sensores para el registro de estos datos biomédicos. Estos datos se envían al servidor web vía WiFi, gracias a la adición de un módulo WiFi a la placa Arduino. Un archivo .php se encarga de la recepción en el servidor, donde se analizan y se almacenan correctamente en una base de datos. Posteriormente los usuarios pueden utilizar la aplicación web desarrollada para consultar y gestionar los datos almacenados. La iniciativa en la comunicación del sistema la toma el sistema de adquisición de datos que será activado mediante botones.

En la siguiente imagen se observa un esquema general de todas las partes implicadas:

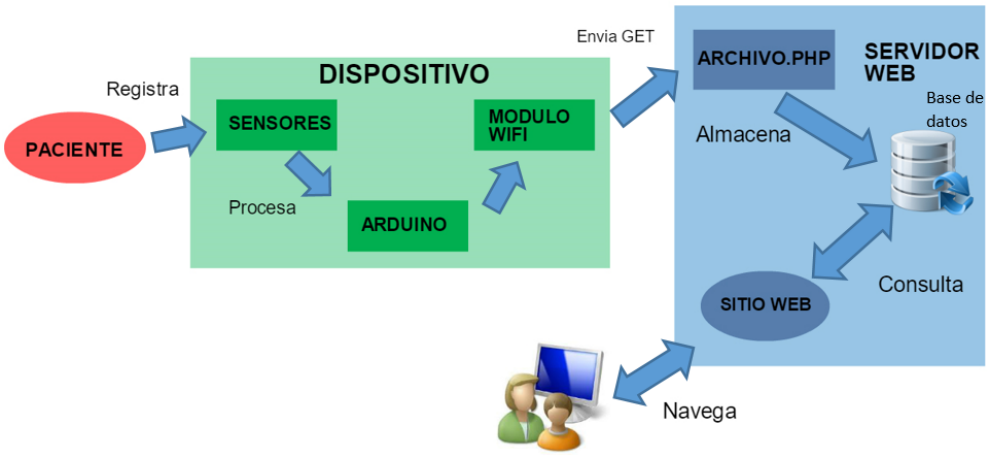


Figura 1. Arquitectura del sistema general

El sistema que se ha diseñado es una unión de distintas herramientas y tecnologías hardware y software, como son:

Software:

- IDE arduino y librerías de comunicaciones.
- Librerías Arduino para Plataforma e-Health V2.0
- PHP, JavaScript, HTML como herramientas de desarrollo web.
- APACHE. Servidor HTTP.
- SQLite como gestor de bases de datos.

Hardware:

- Ordenador Pc o compatible.
- Arduino UNO
- Plataforma e-Health V2.0 para arduino y Raspberry-pi
- Kit de sensores para medidas biomédicas para Plataforma e-Health V2.0

- “Communication Shield” para Arduino
- Módulo WiFi para Arduino: "Roving RN-XVee"
- Router WiFi
- Herramientas eléctricas (Protoboard, botones, leds, resistencias y cables).

2.1. Sistema de captura de datos biomédicos

El sistema de captura de datos biomédicos se compone por la placa Arduino UNO, la plataforma eHealth, la “communication shield” y el módulo WiFi RN-XV 171. Además se hace uso de una protoboard con botones, leds, resistencias y cables para completar todo el sistema.

Arduino

Arduino es una compañía de hardware de código abierto, que se dedica al desarrollo de placas que integran un microcontrolador, con entradas y salidas, analógicas y digitales, en un entorno de desarrollo (IDE).

Arduino llama la atención por su facilidad de uso y su bajo coste, es por ello, por lo que es una herramienta muy utilizada para todo tipo de proyectos, sin necesidad de ser un experto en el manejo. Lo suelen usar alumnos, profesores y cualquier persona que se quiera iniciar en el mundo de la programación. Además la placa de Arduino tiene la capacidad de aumentar su funcionalidad, ya que sobre ella se pueden conectar mediante pines otras placas de extensión, ofreciendo nuevas características y funciones. También se le puede añadir diferentes actuadores y sensores para poder interactuar con el entorno.

Aproximadamente desde el 2005, Arduino hace uso de microcontroladores Atmel AVR de 8 bits, pero hoy en día éstos coexisten con microcontroladores ARM de 32 bits. Para programar el microcontrolador se conecta la placa a un ordenador y se carga el código en ella a través de una comunicación serial. Una vez que el programa ha sido cargado en la placa, no será necesario estar conectado al ordenador, solo sería necesario conectarlo a una fuente de alimentación externa para que el programa se ejecute.

El software de Arduino se puede descargar gratuitamente desde la página oficial de Arduino. Es un entorno intuitivo y de fácil manejo, basado en Processing y en el lenguaje de programación Wiring.

Actualmente existe una gran variedad de modelos de placas de Arduino, sobre 16 modelos. Gracias a ello podemos elegir un tipo u otro dependiendo de la tarea que queramos realizar. Para ello nos debemos fijar en las siguientes características principales:

- Pines: La cantidad de pines analógicos y digitales que son requeridos para realizar el proyecto. Mientras más pines posea la placa, más cosas se pueden hacer y más costosa será la placa.

- Memoria flash: Almacena el código del programa. Dependiendo del tamaño del programa y la cantidad de constantes y variables que se necesiten utilizar, se necesitará una memoria flash más grande o más pequeña.
- SRAM (memoria estática de acceso aleatorio): es donde el programa almacena y manipula las variables cuando el programa está ejecutándose. Si la SRAM se queda sin espacio, el programa de Arduino fallará de forma imprevista, aunque se compile y se suba a Arduino correctamente la aplicación no se ejecutará o se ejecutara de manera extraña.
- Memoria EEPROM: es un espacio de memoria que puede ser utilizado por los programadores para almacenar información a largo plazo.
- Microcontroladores: Pueden ser de 8 bits (AVR) o de 32 bits (ARM). Los de 32 bits tienen mayor rendimiento, pero para la mayoría de proyectos basta con 8 bits.

Arduino UNO

A pesar de que existen una multitud de placas con distintas características, la más utilizada es la Arduino UNO, ya que posee un buen precio y características equilibradas, que permite desarrollar la mayoría de proyectos. Por eso para llevar a cabo este proyecto se ha hecho uso de Arduino UNO. La siguiente fotografía muestra un esquema de las partes de esta placa:

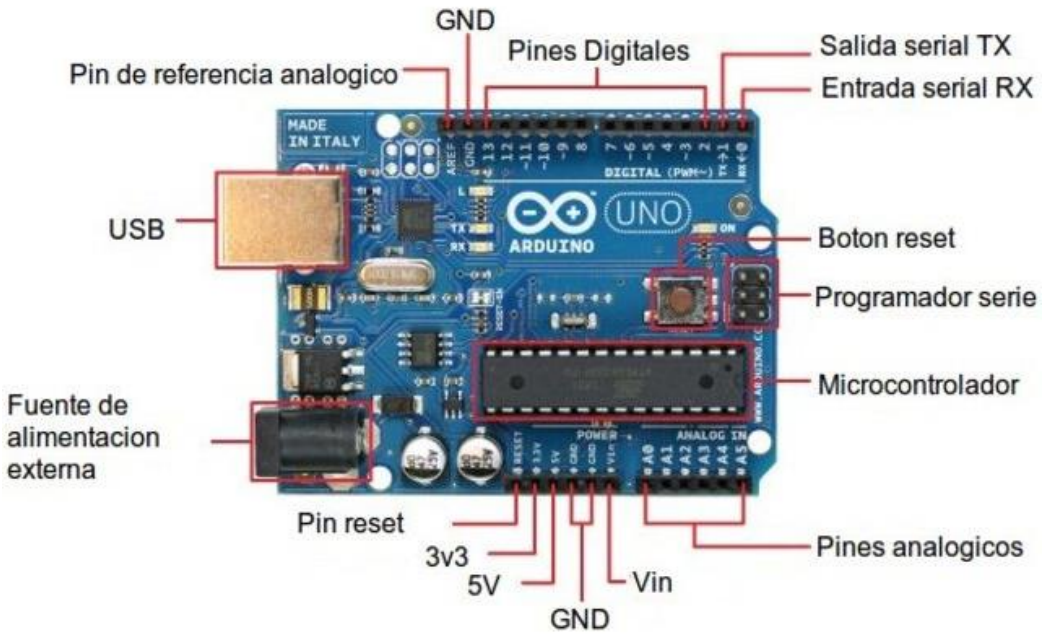


Figura 2. Esquema de las partes del Arduino UNO

Las especificaciones técnicas de la placa Arduino UNO se muestran en la siguiente tabla:

Especificaciones técnicas	
Microcontrolador	ATmega328P
Voltaje Operativo	5V
Voltaje Entrada (recomendado)	7-12V
Voltaje Entrada (límite)	6-20V
Pines E/S digitales	14 (6 proporcionan salida PWM)
Pines Entrada analógica	6
Corriente por pin E/S	20 mA
Corriente pin 3.3V	50 mA
Memoria Flash	32 KB (0,5KB reservados para el bootloader)
SRAM	2 KB
EEPROM	1 KB
Velocidad reloj	16 MHz
Peso	25 g
Dimensiones	68.6 mm x 53.4 mm

Tabla 1. Especificaciones técnicas de Arduino UNO

Módulo WiFi

El intercambio de datos entre la placa Arduino y el servidor web se va llevar a cabo mediante comunicación WiFi. En concreto para este trabajo se ha elegido el módulo WiFi Roving RN-XV 171, principalmente porque cumple con los requisitos para realizar la función que se necesita en este proyecto. Este módulo está desarrollado por Roving Networks y es un sistema de comunicación Wi-Fi especialmente diseñado para migrar de la arquitectura 802.15.4 al estándar TCP/IP sin tener que rediseñar el hardware.

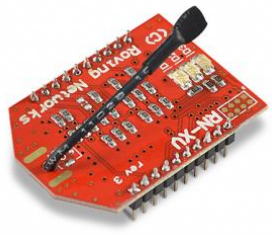


Figura 3. Módulo WiFi Roving RN-XV

Este módulo incorpora un radio 802.11b/g, un procesador de 32 bits, un reloj de tiempo real, una unidad de manejo de potencia y una interfaz análoga. El módulo tiene pre-grabado un firmware que

simplifica la integración y minimiza el tiempo de desarrollo teniendo una simple configuración de hardware donde solo se tienen cuatro conexiones: PWR, TX, RX y GND.

Algunas de las características de este módulo son las siguientes:

- 8Mbit de memoria flash y 128 KB de RAM
- 8 GPIO (Pines genérico cuyo comportamiento se puede controlar por el usuario en tiempo de ejecución) y 3 análogos.
- Puede ser alimentado desde una fuente de 3.3 V DC regulada o de 2 a 3 V desde baterías.
- Seguridad de autentificación WiFi: WEP-128, WPA-PSK (TKIP), WPA2-PSK (AES).
- Estándar: cliente DHCP, cliente DNS, ARP, ping ICMP, FTP, TELNET, HTTP, UDP, TCP.
- Interfaces Hardware: TTL UART
- Host data rate: hasta 464Kbps (UART)
- Posee configuración sobre UART o una interfaz inalámbrica usando comandos simples de ASCII.

Para usar este módulo WiFi hace falta utilizar también la placa “Communication Shield”.

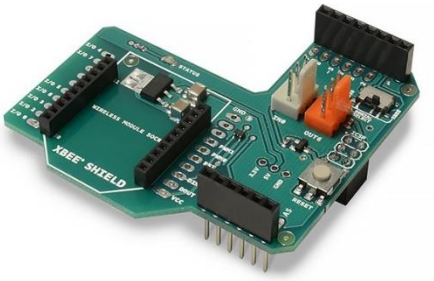


Figura 4. "Communication Shield" para Arduino

Esta placa junto con el módulo WiFi se coloca encima de la placa eHealth, quedando con el siguiente aspecto:

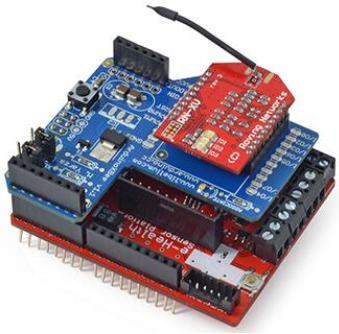


Figura 5. Placa eHealth junto a la placa de comunicación y el módulo WiFi

La placa “Communication Shield” tiene un pequeño interruptor con dos posiciones disponibles: una llamada XBEE y otra llamada USB. El esquema de las comunicaciones según la posición en la que se encuentre el interruptor es el siguiente:

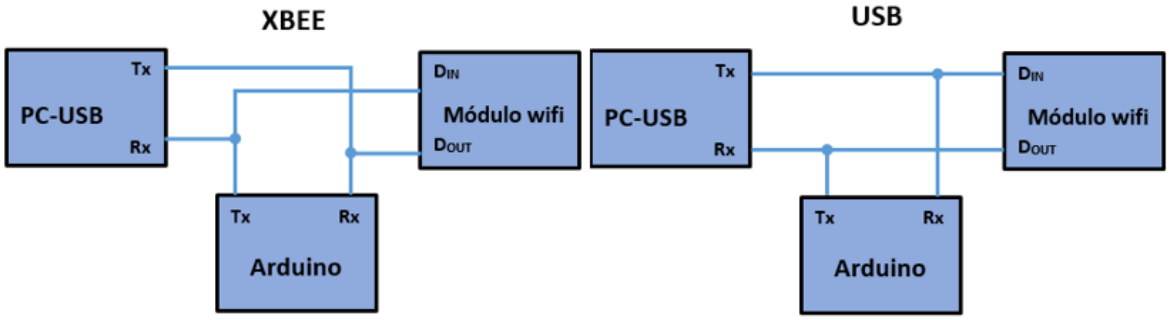


Figura 6. Diferencia de comunicación entre XBEE y USB.

En la posición XBEE, la salida del módulo WiFi está conectada al pin de escucha del microcontrolador y la entrada del WiFi a la salida del microcontrolador, pero hay que tener en cuenta que los pines de comunicación del micro siguen conectados al puerto de comunicaciones del Arduino y este al ordenador. En este modo el microprocesador no puede ser programado desde el ordenador, es decir, no es posible subir ningún sketch con código del programa.

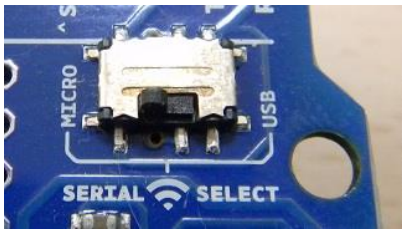


Figura 7. Interruptor en la posicion XBEE.

Hay que poner el interruptor en USB para cargar el código del programa, ya que en esta posición el módulo WiFi está conectado al puerto USB, por lo que se comunica directamente con el PC. Una vez que el programa haya sido cargado en el Arduino, se puede cambiar de posición el interruptor y ponerlo en la posición XBEE. Esta es la forma de cargar los sketchs.

Figura 8. Interruptor en la posición USB

Durante el desarrollo del proyecto se deshabilitó la conexión física del puerto de comunicaciones entre la “Communication Shield” y el Arduino y se creó un canal serie software de comunicaciones entre el micro y el módulo WiFi. Haciendo esto, se evita tener que estar cambiando de posición el interruptor, debido a que el puerto serie se comunica con el Arduino y el Arduino se comunica con el módulo WiFi por otro puerto serie diferente por lo que es posible usar el puerto serie nativo para depuración del código.

Para hacer esto solo se tiene que doblar los pines 0 y 1 de nuestro shield para que no entren en contacto con Arduino e inhabilitar la comunicación serie del Shield con Arduino. La siguiente tarea es crear un canal virtual entre los pines 8 y 9 de Arduino. Para realizar esto es necesario utilizar la librería SoftSerial. El interruptor debe estar en la posición XBEE porque lo que se quiere conseguir es que el módulo WiFi se comunique con el microprocesador para intercambiar comandos.

Así, el Arduino dispondrá de dos puertos serie: el nativo USB qué se usará para programar y para enviar información de depuración y el puerto virtual para la comunicación entre el microprocesador y el módulo WiFi.

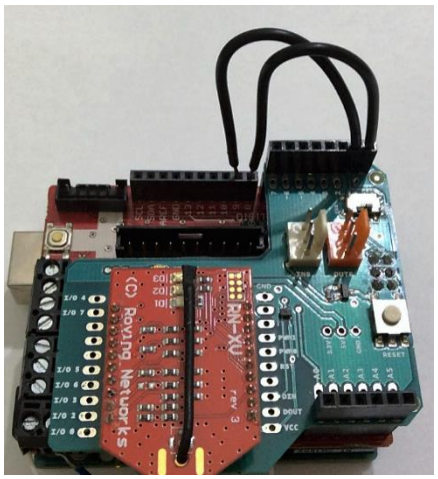


Figura 9. Conexión de los pines para la apertura del puerto virtual.

Una vez listo el hardware, solo queda programar en el código de Arduino la apertura de un nuevo puerto virtual usando la librería “SoftwareSerial”.

```
#include <SoftwareSerial.h>
SoftwareSerial wifiSerial(8,9);
```

Figura 10. Creación del puerto virtual en el programa Arduino.

Finalmente, una vez que se ha terminado la depuración del código, se retira el puerto virtual, porque ya no hace falta.

Antes de empezar a programar, hay que saber que el módulo WiFi puede estar en dos estados: modo comandos o en modo de datos. En el modo de datos el módulo WiFi está preparado para iniciar conexiones externas o aceptar conexiones entrantes. Sin embargo para configurar el modulo hay que estar en modo comandos.

Por defecto el dispositivo estará en modo de datos. Para entrar al modo comandos hay que enviar la siguiente secuencia de tres caracteres: "$$$" y el dispositivo responderá enviando "CMD" indiciando que está en modo Comandos.

Una vez que se está en este modo, es posible enviar secuencias de comandos para configurar el módulo. El dispositivo aceptará bytes ASCII como comandos. Cada comando terminará con un retorno de carro. Para salir del modo Comando, se envía: "exit". El dispositivo responderá con "EXIT" indicando que este ha salido correctamente del modo comando y cambiado a modo Datos. Para la mayoría de los comandos que se envíen, el dispositivo responderá con un "AOK" si son válidos y aceptados o con un "EER" si son inválidos o ha habido algún error.

La sintaxis de los comandos es muy sencilla. Cada comando está compuesto por una palabra clave y varios parámetros adicionales separados por espacios. Los comandos se pueden clasificar en 5 tipos:

- Comandos de Ingreso (SET): modifica los parámetros del módulo. Tienen efecto de inmediato y para que sean permanentes hay que guardarlos.
- Comandos de Petición (GET): Muestra información sobre el módulo y sus parámetros.
- Comandos de Estado (STATUS): Ve que está pasando con la interfaz, estado de IP, etc.
- Comandos de Acción (ACTION): Permite acciones tales como escanear, conectar, desconectar, etc.
- Comandos de Entrada/Salida de Archivos (FILE IO): Actualiza, carga y guarda configuraciones, borra archivos, etc.

Para que las modificaciones sean permanentes hay que enviar el comando "save" para guardarlo, sino el módulo cargará las configuraciones que tenía antes de encender el dispositivo.

Algunos de los comandos más importantes y que se han utilizado para este trabajo son los siguientes:

- $$$: Entra en modo comando.
- Set wlan phrase <string>: introduce la contraseña de la red WiFi (WPA y WPA2).

- Set wlan join <value>: Configura la política para automáticamente unirse/asociarse a una red. Existen 5 posibles valores para este comando:

Valor	Política
0	Manual, no intenta conectarse automáticamente
1	Trata de unirse al punto de acceso que coincide con la SSID y clave almacenada.
2	Se une a la red que tenga más potencia y tenga guardada su clave, sin importar su SSID
3	Reservado, no usado.
4	Crea una red Adhoc, usando la SSID, ip y máscara de red almacenada.

Tabla 2. Política de conexión a una red.

- join <ssid>: une el módulo WiFi a la red con este nombre (si tiene almacenada su clave).
- Set ip host <IP address>: Configura la dirección IP del host, poniendo la del ordenador que sustenta el servidor web.
- Set ip remote <port number>: Configura el número de puerto que el host está escuchando.

Plataforma eHealth

La plataforma eHealth ha sido diseñada por Cooking Hacks con el objetivo de ayudar a quienes están interesados en el registro de datos biomédicos, ya sea por entretenimiento o por investigación y desarrollo. La principal ventaja es su bajo coste comparado con los aparatos de medidas que hoy en día se conocen. Sin embargo, no se puede usar para monitorizar a pacientes críticos, ya que no dispone de las certificaciones médicas oportunas.

Esta plataforma se compone de una placa y de distintos sensores que se utilizan para medir diferentes variables biomédicas. La información que se toma de los sensores puede ser monitorizada a tiempo real o almacenada para su posterior consulta. Además estos datos se pueden enviar de forma inalámbrica utilizando tecnología como WiFi, bluetooth, etc.

Placa eHealth

La placa eHealth es el hardware principal de la plataforma, ya que se encarga de conectar todos los sensores y de recoger todos los datos que estos reciben. La placa eHealth se comunica con Arduino a través de los pines que posee. A parte de los pines, posee un "jumper" de EMG/ECG, un botón de GLCD, un conector para la pantalla LCD y diversos conectores para los distintos sensores.

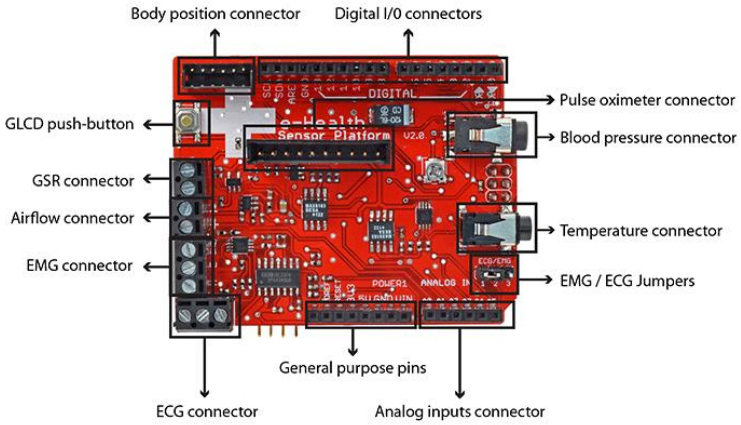


Figura 11. Vista desde arriba de la placa eHealth.

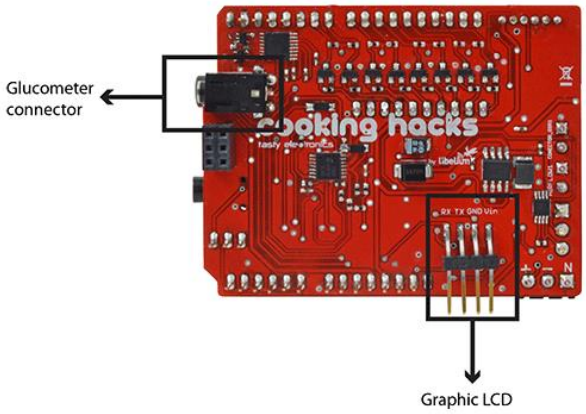


Figura 12. Vista desde abajo de la placa eHealth.

Sensores

En esta plataforma de Salud se pueden hacer uso de 9 sensores distintos. Cada uno proporciona datos de una variable fisiológica que resultará útil para evaluar el estado del paciente. Los sensores que se pueden conectar a la placa eHealth son los siguientes:

- Pulsioxímetro.
- Electrocardiograma (ECG).
- Sensor de flujo de aire en la respiración.
- Sensor de temperatura corporal.
- Sensor de presión sanguínea.
- Sensor de posición del paciente.
- Sensor de conductividad de la piel.
- Glucómetro.
- Electromiograma (EMG).

Figura 13. Placa eHealth con todos los sensores disponibles.

A pesar de esto, para simplificar el proyecto, solo se ha hecho uso de tres de ellos:

- Sensor de temperatura corporal.
- Pulsioxímetro.
- ECG.

La razón de esta elección es que cada uno de ellos representa una modalidad de adquisición y transmisión de datos, lo que permite utilizarlos como modelo, o ejemplo, para una posterior ampliación del número de sensores a contemplar. Tanto el sensor de temperatura como el de electrocardiograma, registran datos analógicos (digitalizados en Arduino) que proviene directamente de la circuitería de amplificación y acondicionamiento. Sin embargo, mientras que el caso del sensor de temperatura es ejemplo de adquisición y transmisión de un solo dato, el del sensor de electrocardiograma permite explorar el caso de la adquisición y transmisión de una señal dada como una secuencia de muestras.
Por su parte la información proporcionada por el pulsioxímetro es digital, proviene de un instrumento comercial, y el sistema de monitorización la lee a través de sus entradas digitales y la transmite. Además, el sistema debe controlar el interfaz con el dispositivo, y sincronizar la lectura, ya que éste posee su propio interfaz de control para activarlo.

Temperatura corporal

La medida de la temperatura corporal es una de las medidas que se realiza en casi todas las evaluaciones de los pacientes, debido a su facilidad para obtenerla, ya que se trata de un método no invasivo y a la gran información que nos puede proporcionar sobre el estado del paciente.

Figura 14. Sensor de temperatura.

La temperatura del cuerpo es útil para detectar la existencia de enfermedades, ya que comprueba la eficiencia del cuerpo en su regulación térmica en función de cambios en la temperatura ambiental y la intensidad de actividad física. Si esta regulación no es buena, se deduce que algo puede estar yendo mal en el paciente.

La temperatura corporal normal de un adulto sano ronda los 36.5 – 37.2 ºC. Existen diferentes estados dependiendo de la temperatura que se tenga:

- Hipotermia, cuando la temperatura es inferior a los 36 o menos.
- Febrícula, cuando la temperatura está entre 37,1-37,9.
- Hipertermia o fiebre, cuando la temperatura es igual o superior a 38.

La temperatura se puede medir en distintas partes del cuerpo, normalmente se mide en axilas, boca o recto. Dependiendo de donde se mida, puede variar un poco el valor que se obtiene.

Hay que tener en cuenta que existen diversos factores que pueden influir sobre la temperatura corporal. Algunos de estos factores son los siguientes:

- Edad del individuo.
- Intensidad de actividad física realizada.
- Constitución del individuo.
- Temperatura y humedad del ambiente.
- Vestimenta.
- Consumo de fármacos.
- Existencia de alguna enfermedad.

Pulsioxímetro

La pulsioxímetría es un método no invasivo que permite controlar la salud de las personas, especialmente aquellas que padecen enfermedades respiratorias o cardiacas crónicas.

Figura 15. Pulsioxímetro.

Este dispositivo se suele colocar en una parte del cuerpo que sea relativamente translúcida, generalmente en los dedos de la mano.

El pulsioxímetro nos da información de dos variables fisiológicas: el pulso y la saturación de oxígeno en sangre. La saturación de oxígeno está definido como la medida de la cantidad de oxígeno disuelto en la sangre, basado en la detección de hemoglobina y oxihemoglobina (HbO2). Esto es posible a que estos compuestos tienen diferentes niveles de absorción de las diferentes longitudes de onda. Se usan dos longitudes de ondas: 660 nm (luz roja) y 940 nm (luz infrarroja). En el lado opuesto el fotorreceptor percibe la luz no absorbida de los dos leds y calcula la saturación de oxigeno arterial.

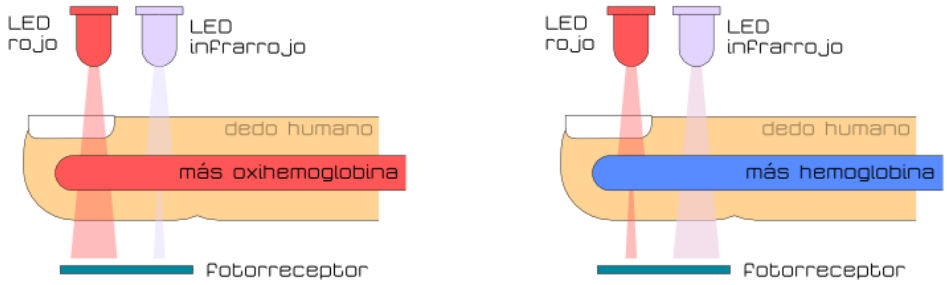


Figura 16. Comportamiento de la luz, de diferentes longitudes de onda, a distintos niveles de hemoglobina en sangre.

Los valores normales de oxígeno en sangre de un individuo sano se sitúan entre 95% y 99%. En cuanto la frecuencia cardiaca en un individuo sano, oscila entre 60 y 100 pulsaciones por minuto. Valores inferiores a 60 pulsaciones por minuto daría lugar a una bradicardia y superiores a 100 a una taquicardia.

Electrocardiograma

El electrocardiograma es un examen diagnostico que registra la actividad eléctrica del corazón. Hoy en día es una de las principales herramientas que se utilizan para evaluar el estado y funcionamiento del corazón. Se usa para medir el ritmo cardiaco y la regularidad de los latidos, posibles patologías pulmonares (embolia de pulmón), problemas cardiovasculares (arritmias, pericarditis, etc), trastornos de los iones como el potasio, el magnesio, etc.

Para su realización se coloca los electrodos en la superficie corporal del paciente.

Figura 17. Sensor de electrocardiograma.

La representación normal de un electrocardiograma sigue el siguiente esquema.

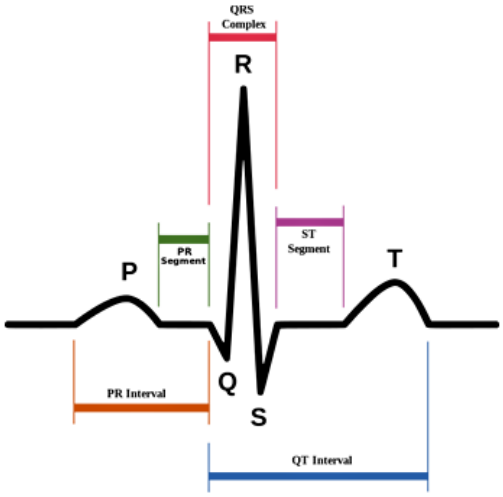


Figura 18. Complejo QRS del electrocardiograma.

El trazado típico de un electrocardiograma registrando un latido cardíaco normal consiste en una onda P, un complejo QRS y una onda T. También existe una pequeña onda U la cual es normalmente inapreciable.

2.2. Sistema de recepción y gestión de datos

El sistema de recepción y gestión de los datos está soportado por un servidor web. En este servidor reside los distintos archivos que hace que todo el sistema funcione correctamente, como son: base de datos, archivo de recepción y aplicación web.

Un servidor web es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o suministrando una respuesta o una información requerida por este. El servidor web sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor, que hablan el uno con el otro utilizando el protocolo HTTP. El término servidor también se emplea para referirse al ordenador que ejecuta el programa.

Nuestro servidor es un servidor web local ya que está alojado en un equipo que está conectado a una red local y solo se podrá utilizar cuando el cliente esté conectado a la misma red local que el servidor. El servidor, se encuentra a la espera de que algún navegador le haga alguna petición, como por ejemplo, acceder a una página web, para responder enviando código HTML mediante una transferencia de datos en red.

Concretamente se ha utilizado un servidor Apache, que es un servidor web HTTP de código abierto. Desde el año 1996, es el servidor web más popular del mundo, debido a su estabilidad y seguridad. Una de las ventajas más grandes de Apache, es que es un servidor web multiplataforma, es decir, puede trabajar con diferentes sistemas operativos y mantener su excelente rendimiento.

Las principales características de Apache son:

- Licencia GPL.
- Soporte de seguridad SSL y TLS.
- Puede realizar autentificación de datos utilizando SGDB.
- Puede dar soporte a diferentes lenguajes, como Perl, PHP, Python y TCL.
- Modular
- Código abierto
- Multi-plataforma
- Extensible

Para instalar Apache hemos hecho uso de XAMPP, que es un servidor independiente de plataforma de código libre que permite instalar Apache sin importar tu sistema operativo. El paquete XAMPP incluye además servidores de base de datos MySQL y SQLite y lenguajes de programación PHP y Perl.

Base de datos

Una base de datos es una estructura en la cual se puede almacenar y organizar datos que comparten entre si algún tipo de vínculo o relación que permita clasificarlos.

Para este proyecto se ha hecho uso de una base de datos para registrar y gestionar toda la información y datos que utiliza este sistema como son: lista de usuarios, lista de pacientes, lista de médicos, lista de dispositivos y lista de sensores.

Esta base de datos es una base de datos relacional, ya que permite establecer interconexiones y relaciones entre los datos. Algunas de las características de estas bases de datos son las siguientes:

- Los datos se organizan en tablas, donde cada tabla a su vez es un conjunto de campos (columnas) y registros (filas).
- La relación entre dos tablas (tabla padre y tabla hijo) se lleva a cabo a través de las claves primarias y claves foráneas.
- La clave primaria es la clave principal de un registro dentro de una tabla y estas identifica de forma única a cada fila de una tabla.
- Las claves foráneas se colocan en la tabla hija, contienen el mismo valor que la clave primaria del registro padre.

Por otro lado la base de datos es dinámica ya que es posible modificar la base de datos en cualquier momento realizando operaciones de inserción, eliminación y modificación de tablas y registros.

En este trabajo se ha utilizado el sistema de gestión de base de datos relacional llamado "Sqlite", concretamente su versión 3. Sqlite es una herramienta de software libre creada por D. Richard Hip en el año 2000. Esta herramienta permite almacenar información de forma sencilla, eficaz y sin requerir un potente hardware. Una de sus principales características es su fácil portabilidad debido a su compatibilidad del 100 % entre las diversas plataformas, por lo que no hay que realizar procesos complejos de importación y exportación de datos.

Otras de sus principales características son las siguientes:

- Está construida en C.
- La base de datos completa se encuentra en un solo archivo.
- Cuenta con librerías de acceso para muchos lenguajes de programación.
- Soporta texto en formato UTF-8 y UTF-16.
- Soportar desde las funciones más básicas hasta las más complejas del lenguaje SQL
- El código fuente es de dominio público.
- En su versión 3, SQLite permite base de datos de hasta 2 Terabyte de tamaño.

Aplicación web

Para la visualización y manejo de los datos se ha creado una aplicación web utilizando principalmente HTML y PHP como lenguaje de programación web. Además se ha hecho uso de distintas herramientas:

- Librería “jpgraph”, para la visualización de gráficas.
- Formularios.
- Imágenes.
- Cifrado “sha1”, para codificar las contraseñas de los usuarios.

La aplicación web accede a la base de datos para consultar los datos que el usuario solicita.

Capítulo 3. Implementación del sistema

3.1. Sistema adquisición de datos biomédicos

Arquitectura del sistema hardware

Para componer la parte hardware del dispositivo, se une la placa Arduino UNO, eHealth, "Communication Shield" y el módulo WiFi, una encima de otra. Los sensores se conectaran a sus respectivos conectores en la placa eHealth. En los pines digitales libres se conectara mediante cables la protoboard con botones y leds. La placa Arduino UNO debe estar conectada a una fuente de alimentación, ya sea un ordenador, una batería externa, etc. En la siguiente imagen se aprecia el esquema del dispositivo.

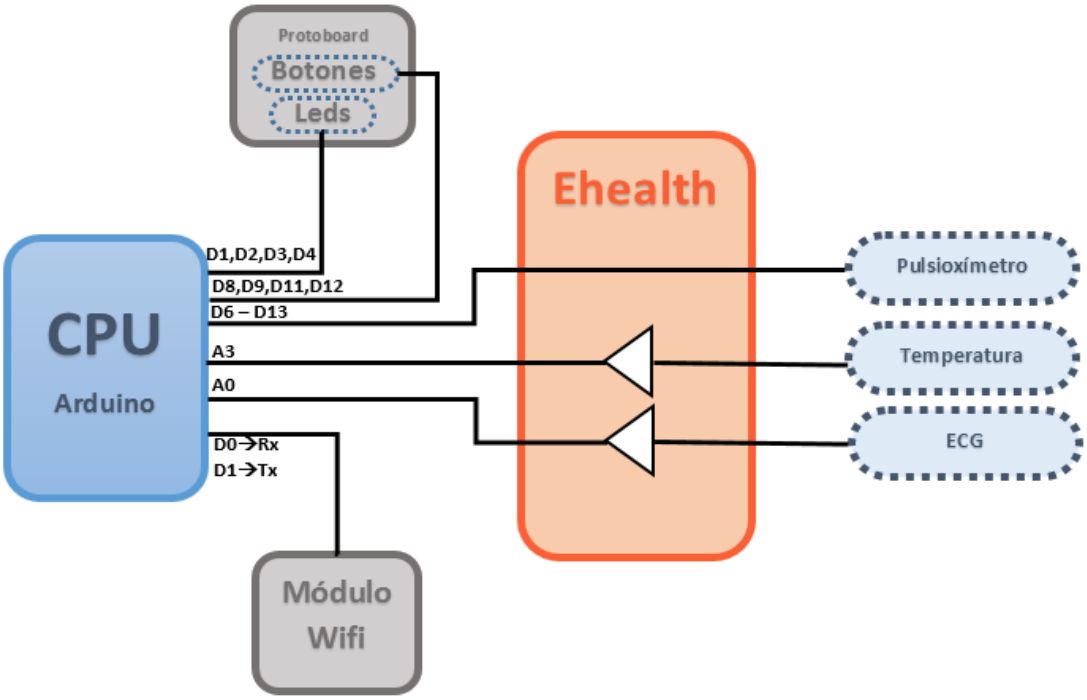


Figura 19. Arquitectura del sistema Hardware

En este proyecto la iniciativa en la comunicación y transmisión de la información la lleva acabo el sistema de monitorización, es por ello que tenemos que diseñar e incluir un cuadro de mando con distintos botones para controlar las distintas funciones de nuestro dispositivo. Para ello se ha hecho uso de varias herramientas electrónicas como son: una protoboard, botones, leds, resistencias y cables.

Se ha puesto un botón para cada sensor utilizado y un botón permite identificar y asignar un paciente concreto a cada dispositivo. Cuando pulse cada uno de estos botones el dispositivo lo reconocerá e interpretara la acción determinada que deberá realizar.

- El botón 1 inicia el proceso de medida y envío de una muestra de temperatura.

- El botón 2 inicia el proceso de medida y envío de las muestras del pulsioxímetro.
- El botón 3 inicia el proceso de medida y envío de una secuencia de ECG.
- El botón 4 lanza una petición al servidor para la asignación y/o modificación del paciente asignado al dispositivo.

A parte de los botones también se ha puesto un led indicativo para cada botón. Estos leds sirven para indicar si el botón ha sido pulsado correctamente, ya que cuando se pulse uno de los botones, el led correspondiente a dicho botón se encenderá y permanecerá encendido o parpadeando según la acción que la CPU esté realizando en cada momento. Una vez que concluya la acción y el dato sea enviado, el led se apagará, indicando que es posible retirar dicho sensor del cuerpo del paciente. El led que está junto al botón de asignación del paciente permanece encendido cuando el dispositivo está disponible para realizar cualquier acción. Si se pulsa este último botón, se encenderá todos los leds, indicando que se está enviando al servidor una petición para asignar un nuevo paciente.

El cuadro de mandos se conecta al Arduino UNO de la siguiente manera:

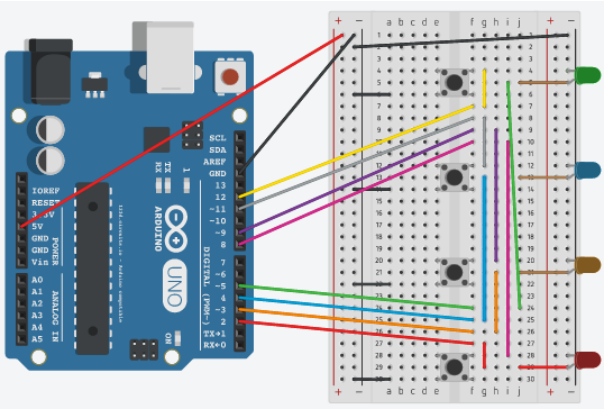


Figura 20. Conexión cuadro de manos con Arduino UNO

Sensores

A continuación se explica los circuitos para cada sensor.

Temperatura corporal

El diseño para el sensor de temperatura se basa en un puente de Wheatstone. El puente de Wheatstone ha sido diseñado para cubrir el rango de temperaturas de interés: entre 25ºC y 50ºC. La tensión de salida diferencial del puente de Wheatstone es amplificada y referenciada a tierra mediante un amplificador de instrumentación. Este sensor se comunica mediante el pin analógico 3 de Arduino. En la siguiente imagen se puede observar el circuito relacionado de este sensor:

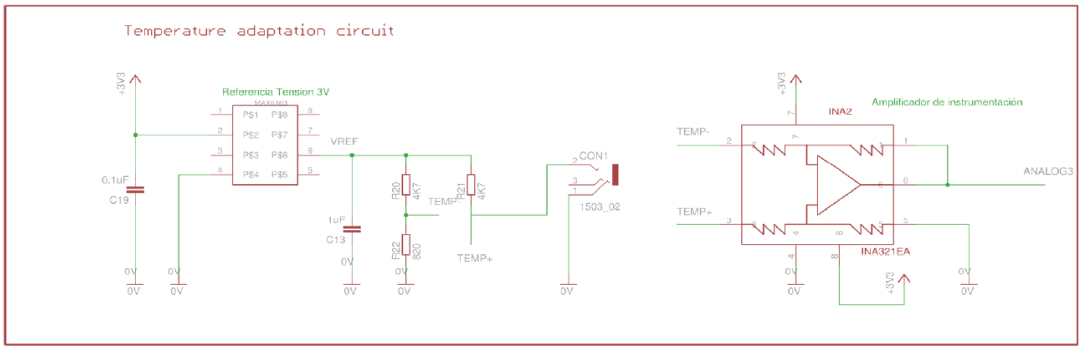


Figura 21. Circuito de adaptación del sensor de temperatura.

Electrocardiograma

La electrónica de adaptación necesaria para el electrocardiograma, ha sido basada en las especificaciones técnicas de un amplificador de instrumentación. Este dispositivo es el encargado de amplificar la señal diferencial de entrada proveniente de los electrodos izquierdo y derecho. Se utilizan amplificadores operacionales para completar las distintas etapas necesarias para la medición de la señal de ECG. Este sensor utiliza el pin analógico 0 de Arduino para la comunicación. En la siguiente imagen se observa observar el circuito relacionado de este sensor:

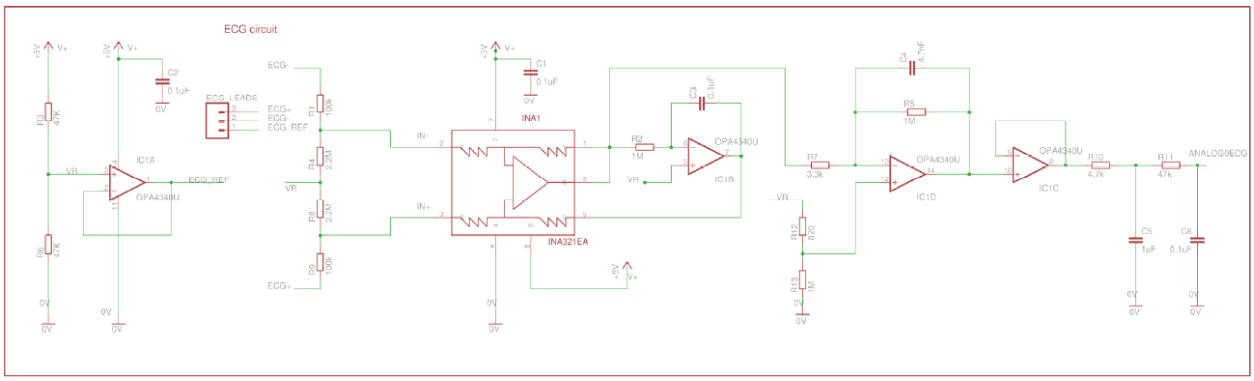


Figura 22. Circuito de adaptación del electrocardiograma.

Pulsioxímetro

Se utiliza etapas transistorizadas para leer los valores que aparecen en la pantalla del dispositivo, utilizando interrupciones como sistema de detección.

Arduino lee la información del pulsioxímetro mediante sus entradas digitales 6 a 13. En la siguiente imagen se puede observar el circuito de interfaz para este sensor:

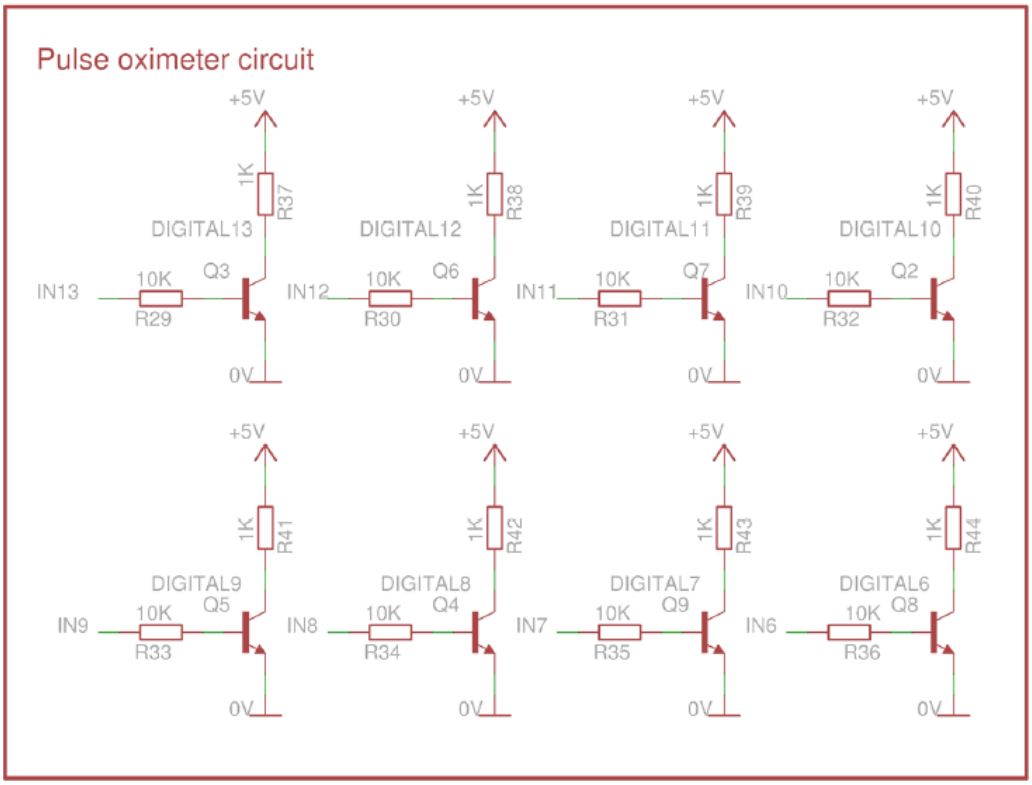


Figura 23. Circuito de adaptación del sensor del pulsioxímetro.

Al mismo tiempo, cuatro de estas entradas (digital 8, 9, 11 y 12) se utilizan para leer el estado de los botones del interfaz de usuario, por lo que la interfaz entre el pulsioxímetro, los botones y la CPU quedaría de la siguiente forma:

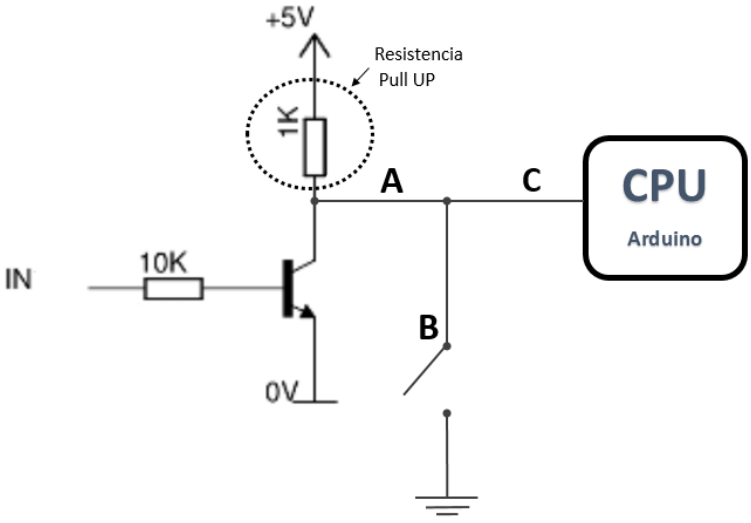


Figura 24. Interfaz entre pulsioxímetro, botón y CPU.

Este circuito representa una AND cableada entre el inversor a la salida del pulsioxímetro y el pulsador, compartiendo la resistencia del Pull Up del inversor con el pulsador. Con esto se consigue que lo que reciba la CPU varíe dependiendo de lo que se reciba del pulsioxímetro y del estado de los botones. Los casos posibles son los siguientes:

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 3. Tabla de verdad del AND.

B = 0, botón pulsado.

A = 1, pulsioxímetro en marcha.

La lectura correcta se controla con el programa de Arduino, ya que este interpreta si lo que se está usando son los botones o el pulsioxímetro en cada caso.

Programa Arduino

El siguiente diagrama de flujo muestra el funcionamiento general del programa completo que se carga en el microcontrolador de Arduino:

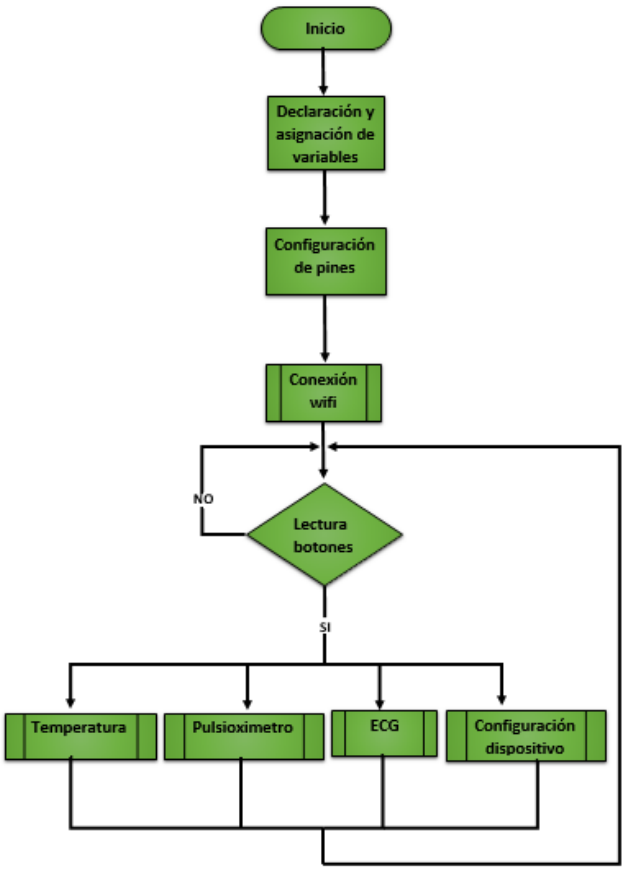


Figura 25. Diagrama de flujo del programa de Arduino.

El funcionamiento del programa se resume en conectar el dispositivo a una red WiFi y hacer la lectura de los botones. Cuando se pulse un botón, se realiza la función que tiene asignado dicho botón. Cuando concluya la acción, se vuelve a la lectura de botones, a la espera de que se pulse de nuevo un botón.

Para la realización del programa se han creado las siguientes funciones:

- Void conexionwifi().
- Void enviarTemperatura().
- Void enviarPulsioximetro().
- Void enviarEcg().
- Void configurarDispositivo().

A este programa se incorporan dos librerías para poder utilizar los sensores. Se puede descargar desde la página principal de CookingHacks, en la sección de e-Health Sensor Platform, mediante un fichero .zip, en el que se incluye tanto la librería de control de los sensores "eHealth" como la librería "PinChageInt" que permite lanzar una interrupción cuando cambia de estado de los pines. Esta última librería tan sólo es necesaria si se va a utilizar el pulsioxímetro dado que necesita de interrupciones.

Conexión WiFi

El diagrama de flujo para la conexión es el siguiente:

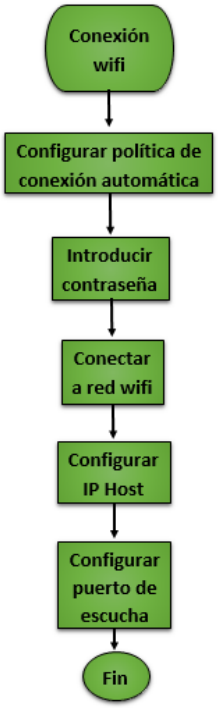


Figura 26. Diagrama de flujo para la función "conexionwifi()".

Toma y envío de datos

Para realizar cualquiera de las funciones disponibles, solo hay que pulsar el botón correspondiente. El dispositivo se encarga de registrar la muestra (en el caso de un botón de un sensor) y de enviar al servidor web una petición GET con la información.

Sensor temperatura

El mecanismo para registrar una temperatura y enviarla al servidor es el siguiente:

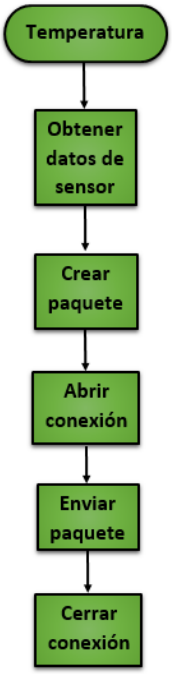


Figura 27. Diagrama de flujo de la función "enviarTemperatura()".

Los datos de temperatura se recogen del sensor gracias a una función proporcionada por la librería eHealth: *eHealth.getTemperature().* Se recoge cuatro muestras de temperatura, una cada segundo. Finalmente la que se envía al servidor es la media de estas cuatro medidas.

Pulsioxímetro

El mecanismo para registrar una muestra de pulso y de porcentaje de oxígeno en sangre y enviarla al servidor es el siguiente:

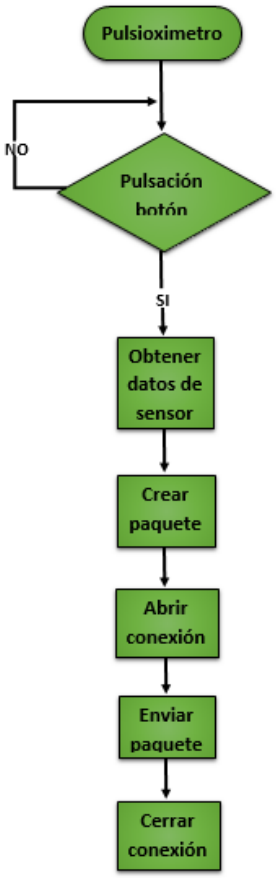


Figura 28.Diagrama de flujo de la función "enviarPulsioximetro()".

El pulsioxímetro necesita encenderse para poder tomar los datos, para ello tiene un botón incorporado en el mismo pulsioxímetro. Una vez encendido, se recogen las muestras gracias a las funciones *eHealth.getOxygenSaturation()* y *eHealth.getOxygenSaturation()* proporcionada por la librería eHealth. Se recogen cuatro muestras de pulso y cuatro de porcentaje de oxígeno en sangre, una de cada por segundo. Finalmente las que se envían al servidor son las medias de cada una.

Electrocardiograma

El mecanismo para registrar una secuencia de ECG y enviarla al servidor es el siguiente:

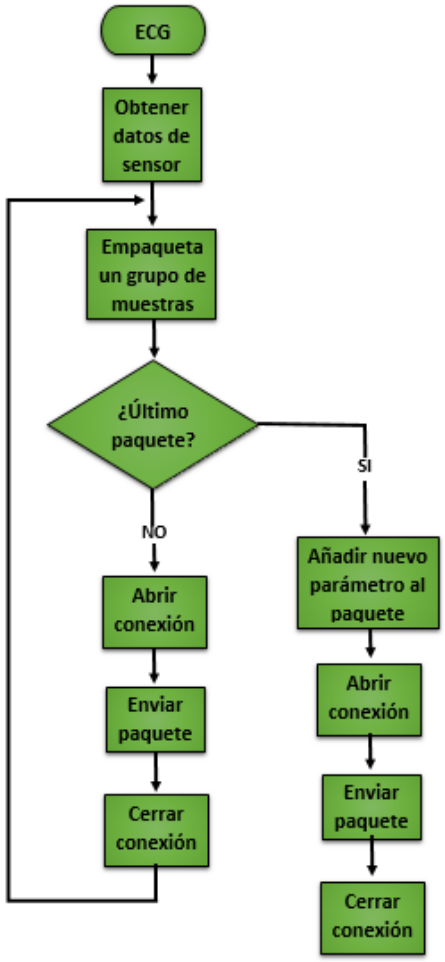


Figura 29. Diagrama de flujo de la función "enviarEcg()".

Para conseguir una secuencia de electrocardiograma se requiere un gran número de muestras. Todas las muestras se deben obtener de manera continuada, con una frecuencia de muestreo lo más grande posible. Los valores de voltaje se recogen gracias a la función *eHealth.getECG()* proporcionada por la librería eHealth.

Para enviar las muestras utilizaremos el formato JSON. Una estructura JSON se compone de dos campos separados por dos puntos de la siguiente forma: {“nombre”: “valor”}. En este caso nuestro campo “nombre” se llama “datos” y nuestro campo “valor” es un vector de las muestras tomadas. Todas las muestras no se pueden enviar en un solo paquete, por lo que hay que fraccionarla y enviarlas en paquetes más pequeños. El primer paquete contendrán la cabecera del formato JSON y el último paquete cerrará la cadena JSON y contendrá también una variable extra que se llama “fin”, que indica que este es el último paquete por enviar.

Número de muestras, tamaño de paquetes y velocidad de transmisión

Se han hecho experimentos para determinar cuál es el número de muestras, tamaño de los paquetes y velocidad de transmisión más adecuados para registrar y transmitir una secuencia de electrocardiograma, teniendo en cuenta las limitaciones del dispositivo.

Lo ideal es enviar el mayor número de muestras posible. En este caso este número está limitado a 295 muestras, ya que el Arduino UNO posee muy poca memoria RAM (2 KB). Sin embargo, hay que tener en cuenta que mientras más muestras se tomen menos memoria quedará disponible para la formación de los paquetes de datos que se deben enviar al servidor web. Además, mientras menos muestras por paquete, más se tardará en enviar toda la secuencia completa, ya que en nuestras pruebas cada paquete tarda aproximadamente 10 segundos en ser enviado al servidor.

La siguiente tabla muestra el mínimo tiempo que se emplea en enviar una secuencia de ECG dependiendo del número de muestras totales considerado. Este número determina el tamaño máximo del paquete y por tanto, el mínimo número de paquetes necesarios para llevar a cabo la transmisión de la secuencia.

Muestras totales	Muestras en paquete	Número de paquetes	Tiempo (mm:ss)
295	1	295	50:00
250	12	21	3:12
200	24	9	1:36
150	37	5	0:54
148	37	4	0:44

Tabla 4. Justificación del número de muestras tomadas para el Ecg.

Frecuencia de muestreo

Según el Teorema de muestreo de Nyquist-Shannon, una señal debe ser muestreada con una frecuencia por lo menos el doble del componente de máxima frecuencia en la señal a muestrear.

Las recomendaciones originales de la AHA (American Heart Association) hablaban de una frecuencia de corte alta de 100 Hz, con lo que una tasa de muestreo de 250 a 300 muestras por segundo resultaría adecuada. Estudios recientes han demostrado que estos datos no son suficientes para la adquisición de ECG en pacientes pediátricos, los cuales requieren que la frecuencia de corte alta sea como mínimo de 250 Hz, con lo que la tasa de muestreo deberá aumentar a un mínimo de entre 750 – 1.000 muestras por segundo. Aun así, para poder realizar un correcto y fiable diagnóstico médico de enfermedades del corazón mediante un electrocardiograma es necesario aumentar la frecuencia de muestreo a 4.000, 8.000 e incluso 15.000 muestras por segundo.

Mientras más muestras por segundo se puedan tomar, más exacto será el electrocardiograma. Al estar limitados por el número de muestras que se pueden tomar, la frecuencia de muestreo no puede ser muy elevada. A modo de ilustración se han hecho pruebas para el caso de secuencias de 148 muestras para determinar qué frecuencia de muestreo es la más adecuada para visualizar al menos varios complejos QRS, para ello se han tomados muestras cada 5, 10, 15 y 20 milisegundos.

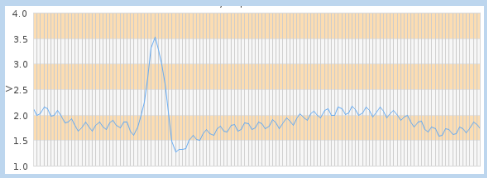
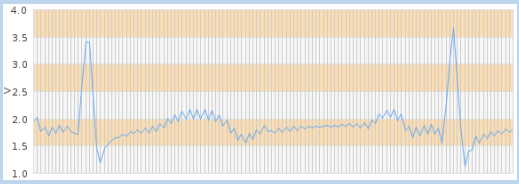
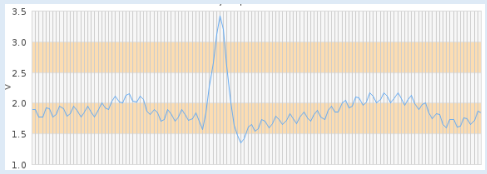
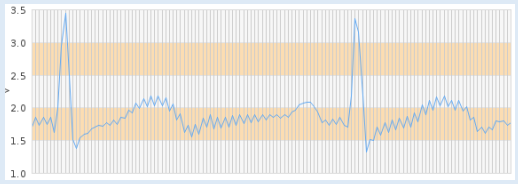
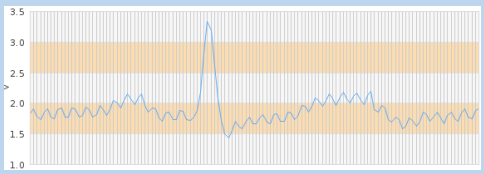
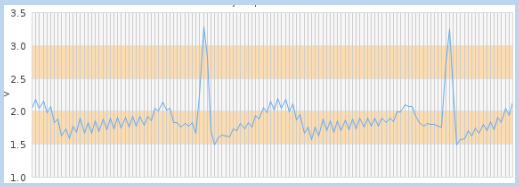
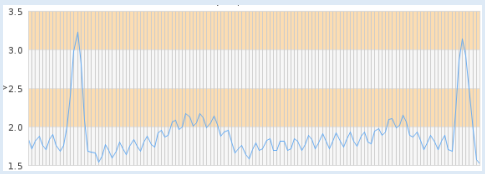
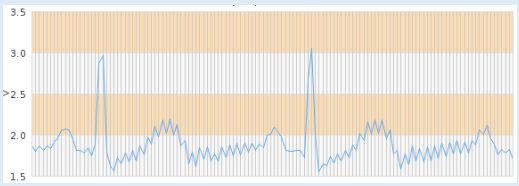
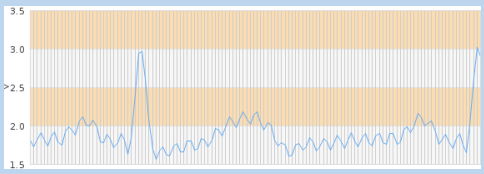
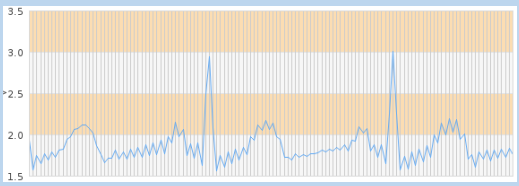
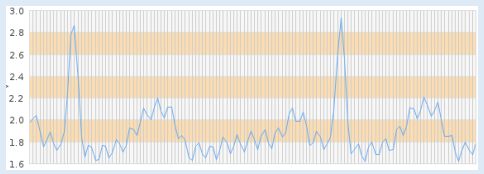
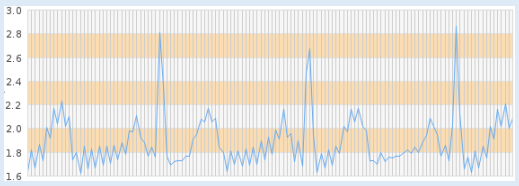
F.M. pulso	5 mseg	10 mseg
60		
75		
90		
105		
120		
150		

Tabla 5. Justificación de la Frecuencia de muestreo utilizada para el Ecg.

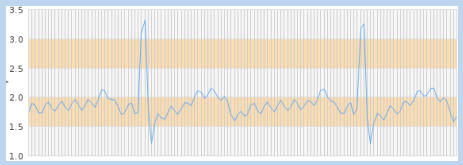
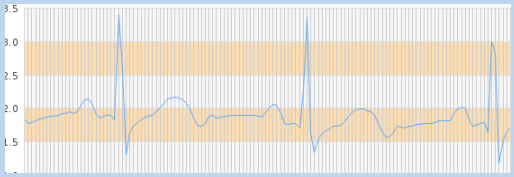
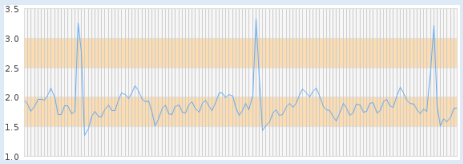
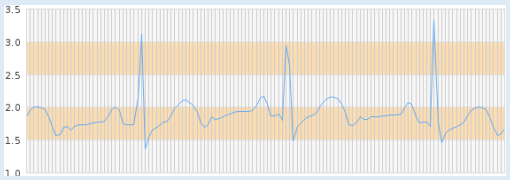
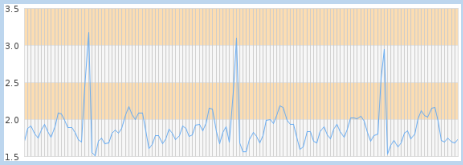
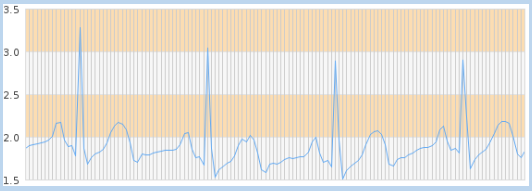
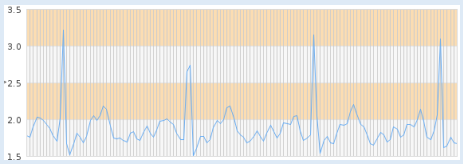
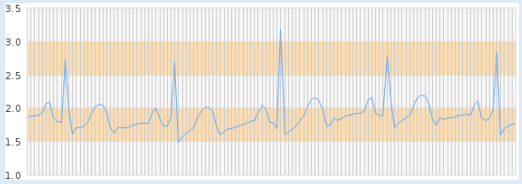
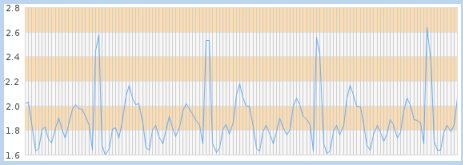
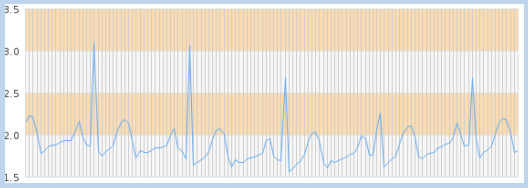
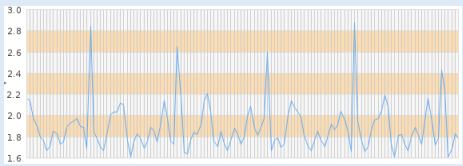
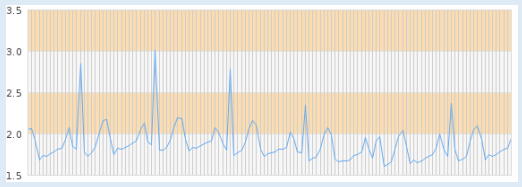
F.M. pulso	15 mseg	20 mseg
60		
75		
90		
105		
120		
150		

Tabla 6. Justificación de la Frecuencia de muestreo utilizada para el Ecg.

Como podemos observar, en este caso los mejores resultados de visualización se obtienen para un periodo de muestreo de 20 ms. Estos registros de electrocardiograma no resultan útiles para realizar un diagnóstico médico, ya que sería necesario obtener una mayor precisión, lo que conlleva un mayor proporcionar un mayor número de muestras. Sin embargo, si nos permite visualizar la forma general que posee un electrocardiograma, e ilustrar el funcionamiento de nuestro sistema.
En la siguiente imagen se muestra las variables que fijan el número de muestras (N), el tamaño máximo de muestras en un paquete (k) y el periodo de muestreo:

```
void enviarEcg(){
  int N=148; //número de muestras.
  int a[N];
  for(int i=0;i<=N-1;i++){
    a[i]=100*eHealth.getECG();        //se convierte en un número entero
    delay(20); //periodo de muestreo
  }
  String dato="{\"datos\":["; //crea cabecera cadena json
  int k=0;

  for (int j=0; j <=N-1; j++){ //recorre todas las muestras
    dato+=a[j];
    k++;
    if((k==37) || (j==(N-1))){ //empaqueta las muestras en paquetes de 37
       if(j==N-1){ //si es la ultima muestra cierra la cadena json
        dato+="]}";
       }
```

Figura 30. Parte de la función "enviarEcg()" que controla el número de muestras y el periodo de muestreo.

Estos valores se pueden modificar en función del tiempo que se esté dispuesto a esperar para enviar todos los datos, siempre y cuando no se exceda el número de muestras y el máximo número de muestras por paquete que se indica en la tabla 4.

3.2. Sistema de recepción y gestión de datos

El sistema de recepción y gestión de datos es el encargado de recibir la información proveniente del sistema de adquisición de datos. Todo está montado sobre un servidor web Apache, que soporta diferentes archivos:

- Archivo “recpeciondatos.php”: es el destino de todos los datos que envía el sistema de adquisición de datos. En este archivo se analiza y gestiona la información que se recibe.
- Base de datos: se almacena toda la información del sistema.
- Aplicación web: para visualizar y consultar la información.

Base de datos

La base de datos se puede crear a través de CMD, accediendo al ejecutable e introduciendo los códigos y comandos, o bien, se puede descargar cualquier aplicación o programa que permita la creación de la base de datos de forma más visual. En este caso, se ha optado por la primera opción.

La base de datos está formada por 7 tablas y sigue el siguiente modelo relacional:

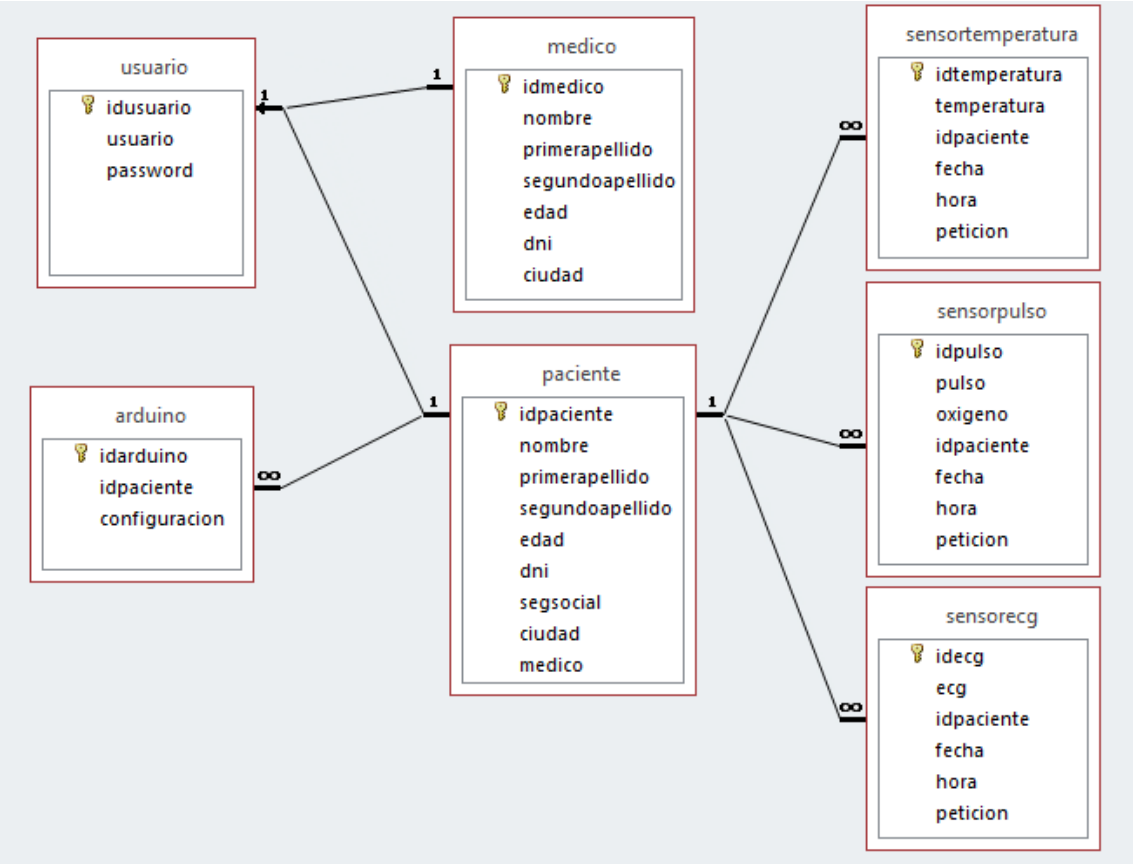


Figura 31. Modelo relacional de la base de datos del sistema.

Las tablas que componen la base de datos son:

- Usuario
- Medico
- Paciente
- Arduino
- Sensortemperatura
- Sensorpulso
- Sensorecg

La tabla Usuarios tiene como objetivo registrar a todos los usuarios que vayan a utilizar la aplicación web. Un usuario será Administrador, Médico o Paciente. Cada usuario tiene un

número único identificativo “idusuario” (este idusuario se añade automáticamente cuando se inserte un nuevo paciente o médico), una identificación del usuario (se utiliza el dni de la persona como identificación excepto en el caso del Administrador que su identificación es “admin) y una clave para acceder a la aplicación web (la clave de cada usuario vuelve a ser su dni, excepto para el Administrador que es “12345”). Dependiendo que tipo de usuario acceda a la aplicación, tendrá unos derechos u otros para poder realizar acciones determinadas. Por un lado el Administrador puede realizar todas las funciones posibles, así como:

- Gestionar la lista de pacientes.
- Gestionar la lista de médicos.
- Gestionar y configurar los dispositivos.
- Gestionar todos los registros medidos.

Por otro lado el usuario Médico puede:

- Ver su propia información.
- Gestionar la lista de pacientes que tiene a su cargo.
- Gestionar los registros de medidas de sus pacientes.

Por último, el usuario Paciente solo podrá ver su propia información y sus muestras registradas. También tiene acceso a la información básica de su médico.

La clave de cada usuario se guardará en la base de dato codificada en SHA1, que se representa por una cadena de 40 dígitos en hexadecimal. Esta cadena es irreversible, es decir, que a partir de esta cadena no se podrá obtener la clave original.

La tabla Médico almacena la información de todos los médicos registrados. Cada médico tiene un numero identificativo “idmedico”. La información básica de cada médico está formada por un nombre, los dos apellidos, la edad, el DNI y la ciudad.

La tabla Pacientes guarda la información de todos los pacientes registrados. Cada Paciente tiene un numero identificativo “idpaciente”. La información básica de cada paciente está formada por un nombre, los dos apellidos, la edad, el DNI, la ciudad y el número identificativo del médico que gestiona a dicho paciente.

La tabla Arduino tiene como objetivo almacenar el id del paciente que va gestionar cada dispositivo. De este modo cada dispositivo tiene un numero identificativo “idarduino”, un campo para el número identificativo del paciente y un campo “configuración” que vale 0, cuando no se haya recibido ninguna petición para asignar un nuevo paciente y 1 cuando si se haya recibido la petición.

La tabla sensortemperatura se encarga de almacenar todos los registros de temperatura que se hagan. Cada registro de temperatura tiene un número identificativo, un campo “temperatura” donde se guarda el valor de la temperatura medida, un campo “idpaciente”

para el numero identificativo del paciente al que se le haya tomado la temperatura, la fecha, la hora y un campo llamado “petición” que vale 0, cuando dicho registro no se haya confirmado y 1 cuando si se haya confirmado.

La tabla sensorpulso se encarga de almacenar todos los registros que se hagan con el pulsioxímetro. Cada registro de pulsioximetría tiene un número identificativo, un campo “pulso” donde se guarda el valor de la frecuencia cardiaca (pulsos por minuto), un campo “oxígeno” donde se guardará el porcentaje de saturación de oxígeno de la hemoglobina en sangre, un campo “idpaciente” para el numero identificativo del paciente al que se le haya realizado la medida, la fecha, la hora y un campo llamado “petición” que vale 0, cuando dicho registro no se haya confirmado y 1 cuando si se haya confirmado.

La tabla sensorecg se encarga de almacenar todos los registros de electrocardiograma que se realicen. Cada registro de ECG tiene un número identificativo, un campo “ecg” donde se guarda un array en formato json con los valores de voltaje que se obtienen, un campo “idpaciente” para el numero identificativo del paciente al que se le haya realizado la medida, la fecha, la hora y un campo llamado “petición” que vale 0, cuando dicho registro no se haya confirmado y 1 cuando si se haya confirmado.

La tabla “Paciente” se relaciona con las tablas “sensortemperatura”, “sensorpulso” y “sensorecg” mediante una relación “uno – muchos” con cada una de ellas, por lo que un paciente tendrá muchos registros de temperatura, pulsioximetría y electrocardiograma. Como hemos mencionado antes, para cada paciente habrá un usuario con el mismo id en la tabla “Usuario”. Ocurre lo mismo con la tabla “Medico”, donde para cada médico habrá un usuario en la tabla “Usuario” que tenga el mismo número identificativo. La tabla que almacena a los dispositivos se relaciona con la tabla de los pacientes, ya que cada dispositivo se encarga de tomar muestras de un paciente determinado.

Para la creación de la base de datos se ha realizado mediante el CMD, introduciendo la siguiente línea: “Sqlite3 nombrebasedatos.db”. Está línea crea una base de datos Sqlite con el nombre “nombrebasedatos” en el directorio que se haya seleccionado. A continuación se crean las tablas con el siguiente comando:

CREATE TABLE “nombre_tabla”(“columna 1” “tipo_de_datos_para_columna_1”, “columna 2” “tipo_de_datos_para_columna_2”,...);

Una vez que se haya creado todas las tablas, ya solo queda ir guardando la información. Para añadir nuevos datos en las tablas se utiliza el siguiente comando:

INSERT INTO “nombre_tabla” (“columna1”, “columna2”, ...) VALUES (“valor1”, “valor2”, ...);

Archivo recepción

Cuando el dispositivo envía información al servidor mediante una petición GET, este la recibe con un archivo llamado “recepciondatos.php”. El diagrama de flujo que representa el funcionamiento de este archivo es el siguiente:

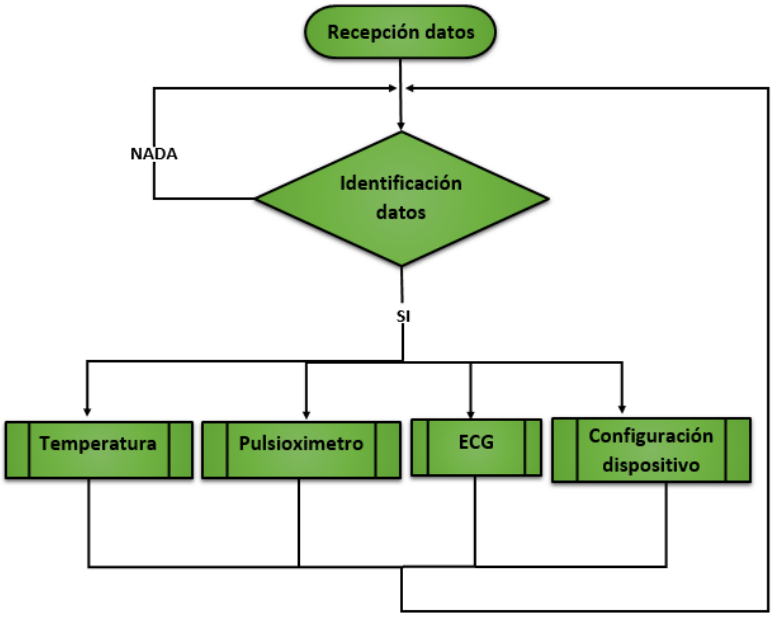


Figura 32. Diagrama de flujo de la recepción de datos en el servidor web.

Este archivo tiene como objetivo recibir la información, analizarla e interpretarla. Una vez que sabe de qué información se trata, la almacenará en la tabla correspondiente de la base de datos.

Esta información puede ser:

- Muestra de temperatura.
- Muestra del pulsioxímetro.
- Secuencia de ECG.
- Petición de asignación de un nuevo paciente al dispositivo.

Dependiendo que tipo de información se reciba se gestiona de una forma u otra.

Temperatura y pulsioxímetro

En el caso de que la información recibida pertenezca al sensor de temperatura o al pulsioxímetro, el procedimiento es el mismo:

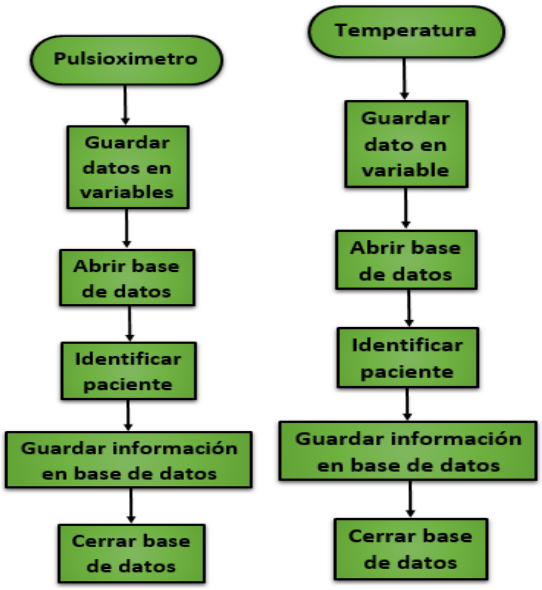


Figura 33. Diagramas de flujo de la recepción de muestra de temperatura y pulsioxímetro.

El mecanismo se resume en guardar los datos recibidos en variables e identificar al paciente, para luego almacenar correctamente el dato en la base de datos.

Electrocardiograma

Debido al tamaño de la secuencia del ECG, es necesario enviar más de un paquete al servidor. Es por ello que se accederá más de una vez a este archivo. El diagrama de secuencias entonces es el siguiente:

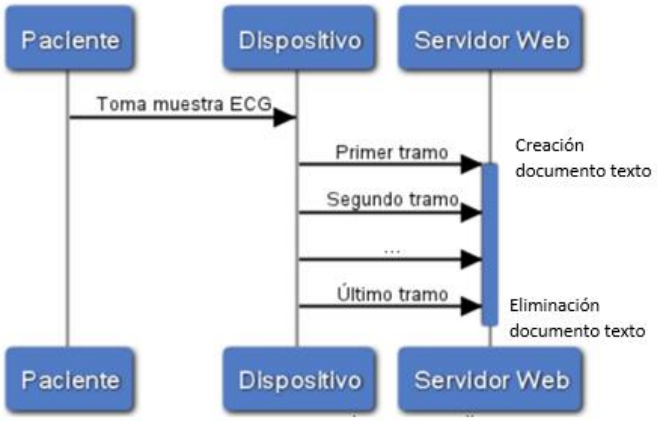


Figura 34. Diagrama de secuencias para la recepción del ECG.

Cada vez que se recibe un paquete en el servidor se lleva a cabo el procedimiento que se indica en el siguiente diagrama de flujo:

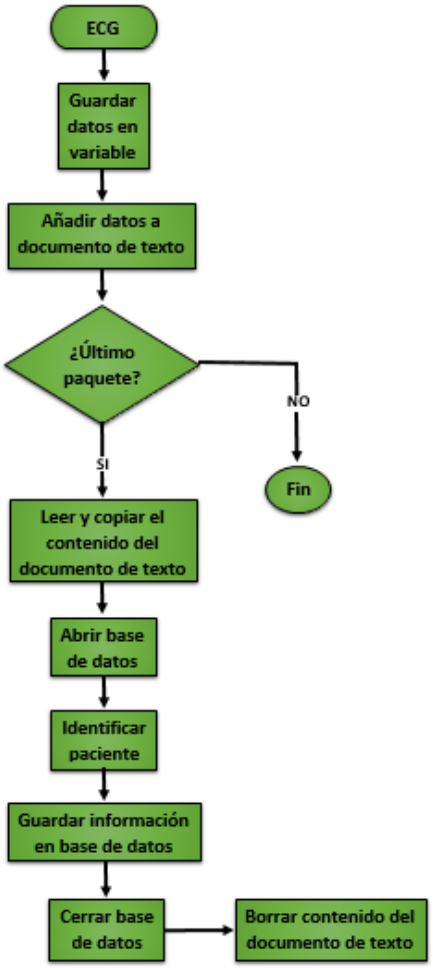


Figura 35. Diagrama de flujo de la recepción de una secuencia de Ecg.

El procedimiento consiste en añadir a un documento de texto cada grupo de muestras que van llegando. Una vez que se identifique el último paquete, se accede al documento de texto y se copia toda la información para almacenarla en la base de datos. Por último, se elimina todo el contenido del documento de texto, para que cuando llegue otra nueva secuencia de ECG, se lo encuentre vacío.

Asignación paciente

Cuando se manda la señal para modificar dicho dispositivo, el servidor la recibe y hace que en la aplicación web aparezca un aviso que indique que ese dispositivo está pendiente de configurar su paciente asignado. El diagrama de flujos es el siguiente:

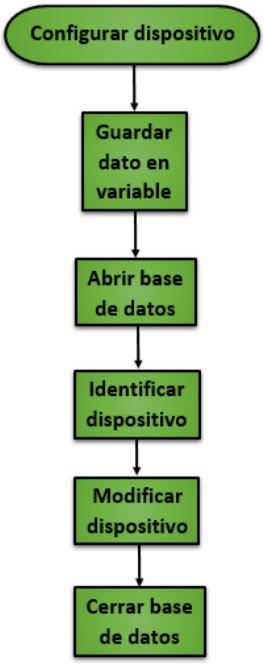


Figura 36. Diagrama de flujo de la recepción de la petición para configurar el dispositivo.

Aplicación web

Se ha diseñado una aplicación web para acceder y consultar los datos almacenados en la base de datos. Esta interfaz de usuario tiene una apariencia muy simple e intuitiva.

Hay tres tipos de perfiles de usuario que pueden acceder a la aplicación. Cada perfil tiene unos derechos y funciones disponibles. Los tipos de usuarios son:

- Administrador.
- Médicos.
- Pacientes.

La arquitectura de la web está representada por la siguiente imagen:

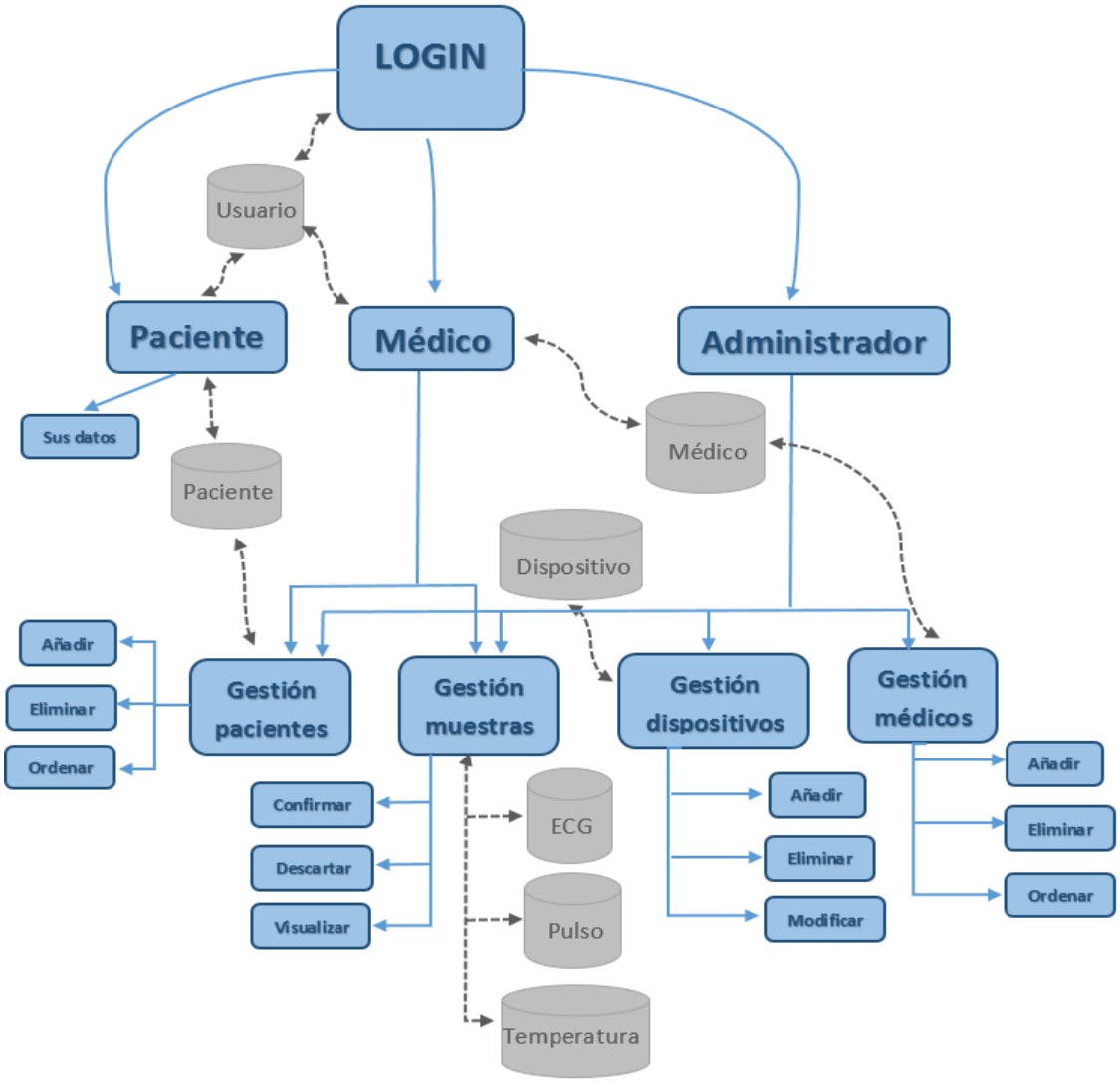


Figura 37. Arquitectura general de la aplicación web.

Para acceder a la aplicación hay que iniciar sesión con un nombre usuario y una contraseña. El sistema reconoce el perfil de usuario que accede a la aplicación y le concede sus derechos.

La página de acceso es la siguiente:

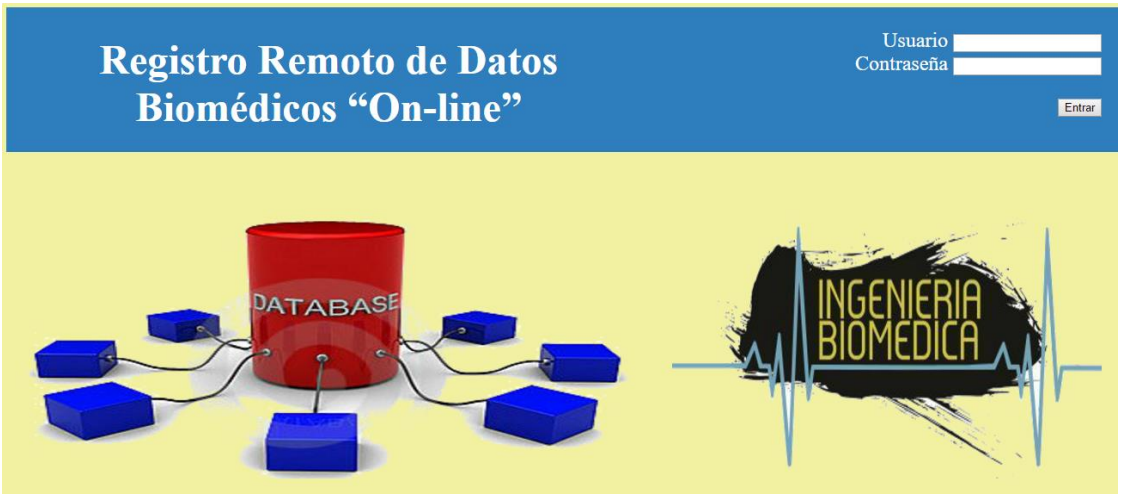


Figura 38. Pantalla de acceso a la aplicación web.

La clave de acceso es cifrada mediante un método de criptografía llamado SHA1. Este método convierte una cadena de texto en otra de 40 caracteres sin importar la longitud de la cadena original, y cifrándola de manera que se hace más difícil poder obtenerla ya que SHA1 no tiene método de reversa para obtener la clave original a partir de una ya cifrada.

Una vez que se inicia sesión, aparece la página inicio de la aplicación donde se puede ver una pequeña descripción del objetivo de la página web. Además informa qué tipo de usuario ha iniciado sesión.

Figura 39. Página inicio de la aplicación web.

La página web se estructura en tres columnas. En la columna de la izquierda aparece el menú principal donde aparece la lista de funciones que dispone cada usuario dependiendo de los derechos que tenga. En la columna del centro, se suele mostrar la lista de los distintos pacientes, médicos, registros, dispositivos, etc. Por último, la columna de la derecha se utiliza para mostrar las gráficas de los registros, así como los botones de añadir y eliminar registros, pacientes, médicos, etc.

Administrador

Hay un único administrador y es el que se encarga de dar soporte a la aplicación web. Tiene todo los derechos posible, así que puede acceder a las distintas listas:

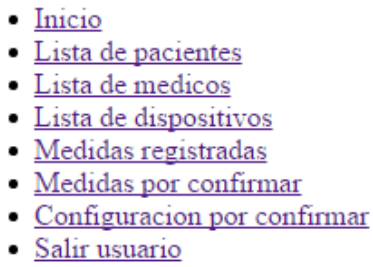


Figura 40. Menú principal dell Administrador.

La lista de pacientes muestra todos los pacientes que han sido dados de alta en la plataforma. Desde aquí se puede añadir, eliminar u ordenar a los pacientes.

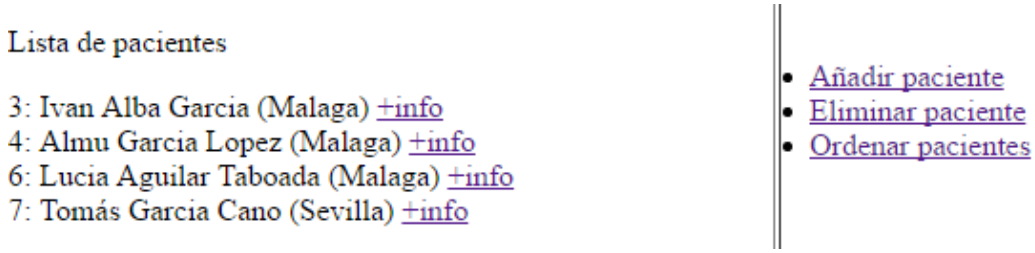


Figura 41. Lista de pacientes.

Tambien se puede acceder a la información de un paciente determinado, donde se muestra su infomación básica junto con los registros que se hayan confirmado de este paciente. A parte de ver el valor numérico de las muestras, se puede ver una gráfica de todos los registros que se han confirmado.

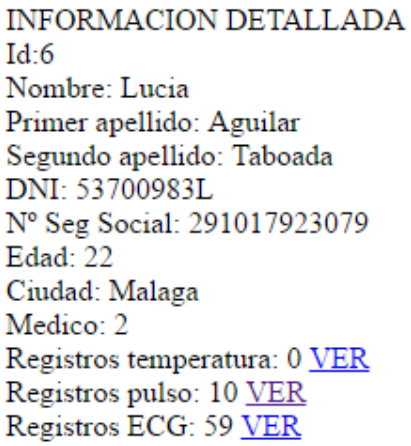


Figura 42. Ejemplo de la información detallada de un paciente.

También es posible acceder a la lista de médicos que se han dado de alta y de igual modo se puede añadir, eliminar u ordenar a estos médicos. Además también se puede acceder a la información de cada médico, donde se puede ver el número de pacientes que tiene a su cargo y acceder a la información de estos.

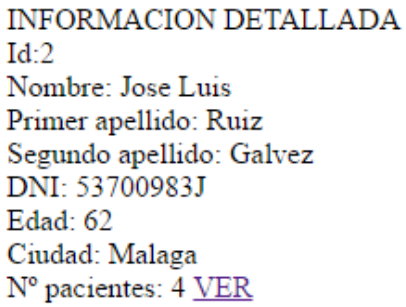


Figura 43. Ejemplo de la información detallada de un médico.

El administrador es el único usuario que tiene acceso a la lista de dispositivo, pudiendo dar de alta o de baja a cualquiera de ellos. Además al administrador le aparecerá un aviso en la esquina inferior izquierda, cuando se solicite una modificación del paciente asignado a un dispositivo. De igual forma, cuando el dispositivo envía una muestra al servidor, le aparece una alerta en pantalla, avisando que hay muestras de sensores por confirmar.

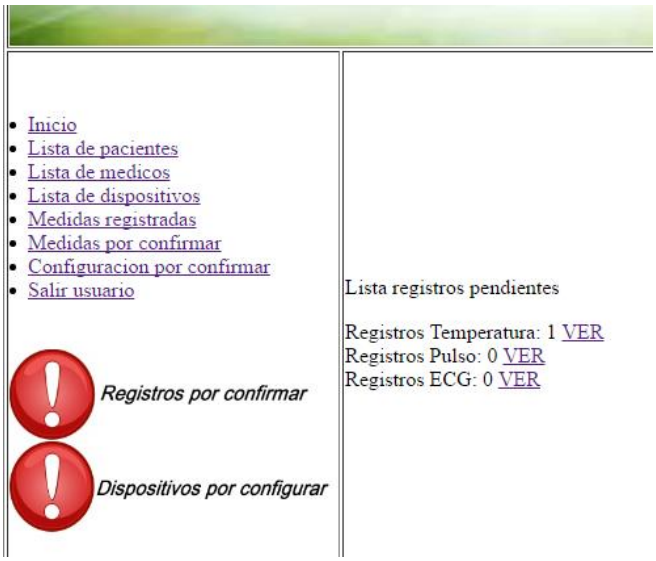


Figura 44. Visualización avisos de nuevos registros y asignación de paciente pendientes.

Médicos

Los médicos tienen acceso a la información de sus pacientes, pudiendo sólo confirmar aquellos registros que pertenezcan a estos.

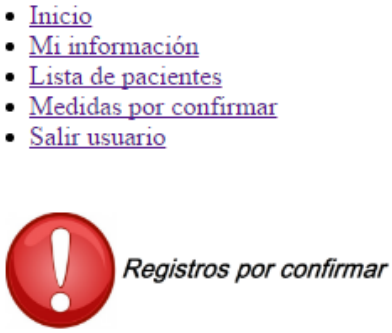


Figura 45. Menú principal para un médico.

Pacientes

Los usuarios que tienen menos privilegios son los pacientes, ya que la única función que tienen disponible es la de ver su propia información. Además pueden ver la información básica del médico que los gestiona.

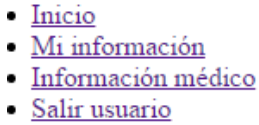


Figura 46. Menú principal para un paciente.

Conclusiones

En este trabajo fin de grado:

- Se ha implementado un sistema de adquisición y registro de datos biomédicos. Para ello:
    - Se ha construido un dispositivo formado por distintos componentes hardware como: Arduino Uno, plataforma eHealth con sus sensores de temperatura, pulsioxímetro y electrocardiograma y la "Communication Shield" junto con el módulo WiFi Roving RN-XV 171.
    - Se ha diseñado y desarrollado el "firmware" encargado de control, adquisición, transmisión e interfaz de usuario.
- Se ha implementado un sistema de recepción y gestión de datos utilizando un servidor web Apache que usa una base de datos y sustenta una aplicación web.
    - Se ha establecido un protocolo de intercambio de información desde el microcontrolador de Arduino hasta el servidor web mediante comunicación WiFi, donde la iniciativa en la comunicación y transmisión de la información la lleva siempre acabo el sistema de adquisición de datos.
    - Se crea un proceso de comprobación y validación de los datos recibidos en el sistema de recepción y gestión de datos.
    - Se ha creado una base de datos que se actualiza conforme llega nueva información al servidor web.
    - Se ha creado un prototipo de aplicación web, que recoger la funcionalidad básica, donde los distintos usuarios pueden visualizar y gestionar la información contenida en la base de datos.

Haciendo balance, se han alcanzado los objetivos que se plantearon al inicio del proyecto.

Líneas futuras

Como líneas futuras que podría dar lugar a la continuidad del trabajo y a la mejora del sistema, se propone:

- Dotar al sistema de adquisición de datos de la memoria suficiente, para poder recoger y transmitir un flujo continuo de muestras de señales.
- Aplicar el campo de medidas fisiológicas incorporando nuevos sensores.
- Estudiar y realizar las modificaciones necesarias para dotar de acceso a internet.
- Diseñar una aplicación web que proporcione mayor funcionalidad y mejore la estética que la del prototipo propuesto.

Bibliografía

[1] Sociedad Internacional para la telemedicina y eSalud. http://www.isfteh.org/working_groups/category/e_hispanic

[2] F.Sebastian et ad.; Real Time E-health System for Continuous Care; Proceedings of the 8th International Conference on Pervasive Computing Technologies for Healthcare, REHAB ICST; pp. 436-439; Brussels, Belgium, 2014 DOI: 10.4108/icst.pervasivehealth.2014.255308

[3] I.Orha, S.Oniga; *Wearable sensors network for health monitoring using e-Health platform;* Carpathian Journal of Electronic and Computer Engineering 7/1 (2014) 25-29

[4] H. Cervantes de Ávila y otros; *Arquitectura de e-Salud basada en redes inalámbricas de sensores*; Revista de Divulgación Científica; Vol. 6, No.2, septiembre-diciembre 2012; ISSN 2007-3585.

[5] W. Chen et ad. ; *Wireless Transmission Design for Health Monitoring at Neonatal Intensive Care Units*; Procceding of the 2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies, 2009. ISABEL 2009.

[6] E. Shanko and M. G. Papoutsidakis*; Real Time Health Monitoring and Wireless Transmission: A Controller Application to Improve Human Medical Needs*; The 4th IEEE International Conference on E-Health and Bioengineering - EHB 2013.

[7] Arduino: https://www.arduino.cc/en/Guide/HomePage

[8] Tutoriales plataforma e-Health, disponibles on-line en:

https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical

[9] Librerias Wifi para Arduino, disponibles on-line en: https://www.arduino.cc/en/Reference/WiFi

[10] Guía de comunicaciones Wifi con arduino, disponible on-line en: http://www.ismsolar.com/blog/la-guia-definitiva-de-comunicaciones-wifi-con-el-arduino-wireless-sd-shield-y-wifly-rn-xv

[11] Roving RN-XVee datasheet, disponible on–line en:

http://www.cooking-hacks.com/skin/frontend/default/cooking/pdf/WiFly-RN-XV-DS.pdf

[12] Roving RN-XVee manual, disponible on–line en:

http://www.cooking-hacks.com/skin/frontend/default/cooking/pdf/WiFly-RN-UM.pdf

[13] Learning PHP, MySQL, and JavaScript, CSS & HTML5. Robin Nixon. Sebastopol, Calif., O'Reilly, cop. 2014

[14] The Essential Guide to HTML5 and CSS3 Web Design. Craig Grannell, Victor S., Berkeley, CA, Apress 2012.

[15] Diseño y Administración de Bases de Datos. Gary W. Hansen. Ed. Prentice Hall. 1997, 2ª Ed.

[16] Guía de comunicaciones WiFi con Arduino y módulo WiFi RN 171, disponible on-line en:

http://www.seeedstudio.com/wiki/Wifi_shield_v1.2#Example_7:_Sending_Data_To_and_Retrieving_Data_From_an_External_Server

Apéndice A: Presupuesto construcción del prototipo

Uno de los principales objetivos del proyecto era diseñar un sistema de bajo coste que pudiera adquirir cualquier persona que quiera registrar sus variables fisiológicas, para conocer el estado de su cuerpo. En el presupuesto solo se han incluido los componentes hardware, ya que las herramientas software no han supuesto ningún coste. Del mismo modo, tampoco se ha incluido el trabajo personal dedicado a la construcción del prototipo.

Articulo	Coste
Arduino UNO [Link]	20 €
Módulo WiFi Roving RN-XV [Link]	40 €
Communication Shield [Link]	15 €
E-Health Platform Shield [Link]	75 €
Sensor de Temperatura corporal [Link]	20 €
Pulsioxímetro [Link]	55 €
Sensor Electrocardiograma [Link]	35 €
Protoboard, botones, leds, resistencias, cables	10 €
TOTAL	270 €

Tabla 7. Presupuesto de construcción del dispositivo.

Como se puede observar se ha conseguido mantener un bajo coste en comparación a los distintos equipos biomédicos que estamos acostumbrados a ver, que suman grandes cantidades de dinero.

Apéndice B: Código Arduino

```
#include <eHealth.h>
#include <PinChangeInt.h>

String ssid="modulo"; //nombre de la red wifi a la que se va conectar
String password="lolaylucky"; //clave de la red wifi a la que se va
conectar
String iphost="192.168.43.223"; //ip del ordenador donde esta
montado el servidor web

char recv[100]; //array que utiliza la función check()
int cont; //variable que utiliza la función check()
int contt = 0; //variable que utiliza la función readPulsioximeter()

//pines que se van a utilizar para los botones
int pushButton12 = 12;
int pushButton11 = 11;
int pushButton9 = 9;
int pushButton8 = 8;
int buttonState12 = 1;
int buttonState11 = 1;
int buttonState9 = 1;
int buttonState8 = 1;

//pines que se van a utilizar para los leds
int led5=5;
int led4=4;
int led3=3;
int led2=2;

void setup()
{
  Serial.begin(9600); //Abre el puerto serie y fija la velocidad en
baudios para la transmisión de datos en serie

  //inicialización de botones
  pinMode(pushButton12, INPUT);
  pinMode(pushButton11, INPUT);
  pinMode(pushButton9, INPUT);
  pinMode(pushButton8, INPUT);

  //inicialización de leds
  pinMode(led5, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led3, OUTPUT);
```

```
  pinMode(led2, OUTPUT);
   //comprobación de que los leds están bien conectados
  digitalWrite(led5, HIGH);
  delay(250);
  digitalWrite(led2, HIGH);
  delay(250);
  digitalWrite(led3, HIGH);
  delay(250);
  digitalWrite(led4, HIGH);
  delay(250);
  digitalWrite(led5, LOW);
  delay(250);
  digitalWrite(led2, LOW);
  delay(250);
  digitalWrite(led3, LOW);
  delay(250);
  digitalWrite(led4, LOW);
  delay(250);

  //inicialización pulsioximetro
  eHealth.initPulsioximeter();
  PCintPort::attachInterrupt(6, readPulsioximeter, RISING);

  conexionwifi(); //función que conecta el dispositivo a la red wifi
}
void loop()
{
  delay(500);

  if(digitalRead(10)){    //solo entra en el IF si el pulsioximetro
no está en marcha
  digitalWrite(led4, HIGH);

  buttonState12 = digitalRead(pushButton12); //lectura boton pin 12
  if( buttonState12 == LOW){ //entra en el IF si está LOW
    digitalWrite(led5, HIGH);
    digitalWrite(led4, LOW);
    enviarTemperatura(); //función que toma muestra de temperatura
y la envia al servidor
    digitalWrite(led5, LOW);
  }

  buttonState11 = digitalRead(pushButton11); //lectura boton pin 11
  if( buttonState11 == LOW){ //entra en el IF si está LOW
    digitalWrite(led2, HIGH);
```

```
    digitalWrite(led4, LOW);
      enviarPulsioximetro();  //función  que  toma  muestra  del
pulsioximetro y la envia al servidor
    digitalWrite(led2, LOW);
  }

  buttonState9 = digitalRead(pushButton9); //lectura boton pin 9
  if( buttonState9 == LOW){ //entra en el IF si está LOW
    digitalWrite(led3, HIGH);
    digitalWrite(led4, LOW);
      delay(5000);
      enviarEcg(); //función  que  registra  un  ecg  y  lo  envia  al
servidor
    digitalWrite(led3, LOW);
  }

  buttonState8 = digitalRead(pushButton8); //lectura boton pin 8
  if( buttonState8 == LOW){ //entra en el IF si está LOW
    digitalWrite(led5, HIGH);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
      configurarDispositivo(); //función que envia una petición para
cambiar el paciente del dispositivo
    digitalWrite(led5, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
  }
  }else{   //si no entra en el IF, sale por el ELSE y apaga el led4
   digitalWrite(led4, LOW);
  }
}
void conexionwifi(){ //creación función conexionwifi()
   while  (Serial.available()>0)  {}  //no  sale  del  bucle  WHILE
mientras haya algo para leer desde el buffer serie,

    //formación de la cadena STRING "join 'nombreRedWifi'\r"
    String conexion="join ";
    conexion+=ssid;
    conexion+="\r";

    //formación de la cadena STRING "set wlan phare 'claveWifi'\r"
    String clave= "set wlan phrase ";
    clave+=password;
    clave+="\r";
```

```
    //formación de la cadena STRING "set ip host 'ipServidorHost'\r"
    String ip= "set ip host ";
    ip+=iphost;
    ip+="\r";
    Serial.print("$$$"); check(); //Entra en modo comando
    digitalWrite(led5, HIGH);
    //Configura la política para automáticamente unirse/asociarse a
una red.
    //EL valor 1 hace que trate de unirse al punto de acceso que
coincide con la SSID y clave almacenada.
    Serial.print("set wlan join 1\r"); check();
    Serial.print(clave); check(); //Introduce la clave de la red
wifi
    digitalWrite(led2, HIGH);
    Serial.print(conexion); check(); //Se conecta a la red wifi
    digitalWrite(led3, HIGH);
    Serial.print(ip); check(); //Configura la ip host, añadiendo la
del servidor Apache.
    digitalWrite(led4, HIGH);
    Serial.print("set ip remote 80\r"); check(); //Configura el
número de puerto en el que el host escucha
    digitalWrite(led5, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
}

void enviarTemperatura(){ //creación función enviarTemperatura()
  delay(1000);
  float temp[4]; //crea array de tamaño 4
  //guarda 4 registros de temperatura, uno cada segundo
  for(int i=0; i<=3;i++){
    temp[i]=eHealth.getTemperature();
    digitalWrite(led5, LOW);
    delay(500);
    digitalWrite(led5, HIGH);
    delay(500);
  }
  float  temperatura  =(temp[0]+temp[1]+temp[2]+temp[3])/4;  //hace
media de los 4 registros tomados
  //crea la cadena de la peticion GET
  String                                                envio="GET
/tfg/gestionpacientesconfiguracion/recepciondatos.php?idarduino=1&
temperatura=";
  envio+=temperatura;
```

```
  Serial.print("$$$"); check(); //entra en modo comandos
  Serial.print("open\r\n"); //abre conexión
  delay(1000);
  Serial.println(envio); //envia peticion GET previamente creada
  delay(1000);
  Serial.print("close\r"); //cierra conexión
}

void       enviarPulsioximetro(){       //creación       funcion
enviarPulsioximetro()
      //no sale del bucle WHILE hasta que se pulse el botón propio
del pulsioximetro
      while(digitalRead(6) && digitalRead(7) && digitalRead(8) &&
digitalRead(9)   &&   digitalRead(10)   &&     digitalRead(12)   &&
digitalRead(13)){
           Serial.println("esperando pulstar boton");
          digitalWrite(led2, LOW);
          delay(250);
          digitalWrite(led2, HIGH);
          delay(250);
      }
      delay(2000);
      //while((digitalRead(10)==0)) {
      while((eHealth.getBPM()==0)                                    ||
(eHealth.getOxygenSaturation()==0)){  //no sale del WHILE hasta que
el pulsioximetro empiece a mostrar medidas
        while((eHealth.getBPM()==0)                                  ||
(eHealth.getOxygenSaturation()==0)) { //no sale del WHILE hasta que
el pulsioximetro empiece a mostrar medidas
            Serial.println("sin numero");
            digitalWrite(led2, LOW);
            delay(250);
            digitalWrite(led2, HIGH);
            delay(250);
            digitalWrite(led2, LOW);
            delay(250);
            digitalWrite(led2, HIGH);
            delay(250);
        }
        delay(2000);
      }

      delay(2000);
      int oxi[4]; //crea array de tamaño 4 para las muestras de
oxigeno en sangre
```

```
      int pul[4]; //crea array de tamaño 4 para las muestras de pulso
      for(int i=0; i<=3;i++){
        pul[i]=eHealth.getBPM(); //toma las muestras de pulso
        oxi[i]=eHealth.getOxygenSaturation();  //toma  las  muestras
de oxigeno en sangre
            digitalWrite(led2, LOW);
            delay(500);
            digitalWrite(led2, HIGH);
            delay(500);
       }
      int  pulso=(pul[0]+pul[1]+pul[2]+pul[3])/4;  //media  muestras
pulso
      int oxigeno=(oxi[0]+oxi[1]+oxi[2]+oxi[3])/4; //media muestras
oxígeno

      //crea la cadena de la peticion GET
      String envio= "GET
/tfg/gestionpacientesconfiguracion/recepciondatos.php?idarduino=1&
pulso=";
      envio+=pulso;
      envio+="&oxigeno=";
      envio+=oxigeno;
      Serial.print("$$$"); check(); //entra en modo comandos
      Serial.print("open\r\n"); //abre conexión
      delay(1000);
      Serial.println(envio); //envia la petición GET al servidor
      delay(1000);
      Serial.print("close\r\n"); //cierra conexión
      delay(5000);
      //no sale del WHILE hasta que se retire el dedo
      while((digitalRead(6)==0    )||    (digitalRead(7)==0)    ||
(digitalRead(8)==0) || (digitalRead(9)==0) || (digitalRead(11)==0)
|| (digitalRead(12)==0) || (digitalRead(13)==0)) {
        Serial.println("retirar dedo");
        digitalWrite(led2, LOW);
        delay(250);
        digitalWrite(led2, HIGH);
        delay(250);
      }
      delay(2000);
}

void readPulsioximeter(){
  contt++;
  if (contt == 50) { //coge solo 50 muestras para reducir la latencia
```

```
    eHealth.readPulsioximeter();
    contt = 0;
  }
}

void enviarEcg(){
  int N=148; //número de muestras.
  int a[N];
  for(int i=0;i<=N-1;i++){
    a[i]=100*eHealth.getECG();          //se convierte en un número
entero
    delay(20); //periodo de muestreo.
  }
  String dato="{\"datos\":["; //crea cabecera cadena json
  int k=0;

  for (int j=0; j <=N-1; j++){ //recorre todas las muestras
    dato+=a[j];
    k++;
    if((k==37) || (j==(N-1))){ //empaqueta las muestras en paquetes
de 40
       if(j==N-1){ //si es la ultima muestra cierra la cadena json
        dato+="]}";
       }
       String                                                envio="GET
/tfg/gestionpacientesconfiguracion/recepciondatos.php?idarduino=1&
ecg=";
       envio+=dato;
       if(j==N-1){
        envio+="&fin=99";  //añade  variable  para  que  el  servidor
reconozca que es el final
       }
       digitalWrite(led3, HIGH);
       Serial.print("$$$"); check();
       Serial.print("open\r\n");
       delay(1000);
       Serial.println(envio);
       digitalWrite(led3, LOW);
       delay(1000);
       dato=",";
       k=0;
    }else{
      dato+=",";
    }
  }
```

```
}

void configurarDispositivo(){
  String                                                    envio="GET
/tfg/gestionpacientesconfiguracion/recepciondatos.php?idarduino=1&
configuracion=1";
  Serial.print("$$$"); check();
  Serial.print("open\r\n");
  delay(1000);
  Serial.println(envio);
  delay(1000);
  Serial.print("close\r");
}

void check(){
cont=0; delay(1000);
// Recibe la respuesta del modulo Wifi.
while (Serial.available()>0)
{
  recv[cont]=Serial.read(); delay(100); //guarda la respuesta
  cont++;
}
recv[cont]='\0';
// Pinta la respuesta en el monitor serial.
Serial.println(recv);
// Limpia el el buffer para permitir enviar un comando.
Serial.flush(); delay(1000);
}
```

Apéndice C: Código Recepción de datos.

```
<?php
      //entra en el IF si se recibe el parámetro "temperatura"
      if(isset($_GET['temperatura'])){
            //se guarda en variables los parámetros recibidos
            $idarduino                                                =
htmlspecialchars($_GET["idarduino"],ENT_QUOTES);
            $temperatura                                              =
htmlspecialchars($_GET["temperatura"],ENT_QUOTES);
            //se abre la base de datos "final.db"
            $db                              =                        new
SQLite3('C:/xampp/htdocs/tfg/gestionpacientesconfiguracion/final.d
b');
            //se consulta a que paciente gestiona este dispositivo
            $sql     =     "SELECT     *     FROM     arduino     where
idarduino=$idarduino";
            $result = $db->query($sql);
            while ($row = $result->fetchArray(SQLITE3_ASSOC)){
                  $idpaciente=$row['idpaciente'];
            }
            //se almacena la fecha y la hora
            $fecha=date("d/m/y");
            $time=date("H:i:s");
            //se  inserta  el  valor  de  temperatura  al  paciente
correspondiente
            $db->exec("insert                                        into
sensortemperatura(temperatura,idpaciente,fecha,hora,peticion)
            values($temperatura,$idpaciente,'$fecha','$time',0)");
            //se cierra la base de datos
            unset($db);

      //entra en el IF si se recibe el parámetro "pulso"
      }else if(isset($_GET['pulso'])){
            //se guarda en variables los parámetros recibidos
            $idarduino                                                =
htmlspecialchars($_GET["idarduino"],ENT_QUOTES);
            $pulso = htmlspecialchars($_GET["pulso"],ENT_QUOTES);
            $oxigeno                                                  =
htmlspecialchars($_GET["oxigeno"],ENT_QUOTES);
            //se abre la base de datos "final.db"
            $db                              =                        new
SQLite3('C:/xampp/htdocs/tfg/gestionpacientesconfiguracion/final.d
b');
            //se consulta a que paciente gestiona este dispositivo
```

```
          $sql    =    "SELECT    *    FROM    arduino    where
idarduino=$idarduino";
          $result = $db->query($sql);
          while ($row = $result->fetchArray(SQLITE3_ASSOC)){
               $idpaciente=$row['idpaciente'];
          }
          //se almacena la fecha y la hora
          $fecha=date("d/m/y");
          $time=date("H:i:s");
          //se  inserta  el  valor  de  temperatura  al  paciente
correspondiente
          $db->exec("insert                                    into
sensorpulso(pulso,oxigeno,idpaciente,fecha,hora,peticion)

     values($pulso,$oxigeno,$idpaciente,'$fecha','$time',0)");
          //se cierra la base de datos
          unset($db);

     //entra en el IF si se recibe el parámetro "ecg"
     }else if(isset($_GET['ecg'])){
          //se  abre  el  fichero  de  texto  y  se  pega  los  datos  a
continuación del último
          $file = fopen("data.txt", "a");
          fwrite($file, $_GET["ecg"]);
          fclose($file);
          //entra en el IF si se recibe el parámetro "fin"
          if(isset($_GET["fin"])){
               //se  lee  el  fichero  completo  y  se  guarda  en  una
variable
               $fp = fopen("data.txt", "r");
               while(!feof($fp)) {
                    $json = fgets($fp);
                    echo $json;
               }
               fclose($fp);
               $idarduino                                          =
htmlspecialchars($_GET["idarduino"],ENT_QUOTES);
               //se abre la base de datos
               $db                    =                    new
               SQLite3('C:/xampp/htdocs/tfg/gestionpacientesconfi
               guracion/final.db');
               //se  consulta  a  que  paciente  gestiona  este
dispositivo
               $sql    =    "SELECT    *    FROM    arduino    where
idarduino=$idarduino";
```

```
                $result = $db->query($sql);
                while ($row = $result->fetchArray(SQLITE3_ASSOC)){
                     $idpaciente=$row['idpaciente'];
                }
                $fecha=date("d/m/y");
                $time=date("H:i:s");
                //se inserta la cadena de datos de ecg al paciente
correspondiente
                $db->exec("insert          into          sensorecg
(ecg,idpaciente,fecha,hora,peticion)
                values ('$json',$idpaciente,'$fecha','$time',0)");
                //se limpia el fichero de texto
                unset($db);
                $file = fopen("data.txt", "w");
                fclose($file);
          }

     //entra en el IF si se recibe el parámetro "configuracion"
     }else if(isset($_GET['configuracion'])){
          $idarduino                                                =
htmlspecialchars($_GET["idarduino"],ENT_QUOTES);
          $db                          =                          new
SQLite3('C:/xampp/htdocs/tfg/gestionpacientesconfiguracion/final.d
b');
          //se actualiza la variable configuracion del dispositivo
con la identificacion obtenida
          $db->exec("UPDATE  arduino  SET  configuracion=1  WHERE
idarduino='$idarduino'");
          unset($db);
     }else{
     }
?>
```

Apéndice D: Árbol de carpetas y archivos de Aplicación Web

```
|   data.txt
|   final.db
|   recepciondatos.php
|
+---acceder
|   |   formulario.php
|   |   login.php
|   |   user.txt
|   |
|   \---imagenes
|           basedatosecg.png
|           basesinfondo.png
|           ingenieriabiomedicalogo.jpg
|           ingenieriabiomedicalogo.png
|
+---imagenes
|       basedatosecg.png
|       dispositivosporconfigurar.png
|       ingenieriabiomedicamanos.jpg
|       registrosporconfirmar.png
|
\---usuarios
    +---admin
    |       configuracionpeticiones.php
    |       leerbasedatosecg.php
    |       listaarduinos.php
    |       listamedico.php
    |       listapaciente.php
    |       medidaspeticiones.php
    |       medidasregistradas.php
    |       paginainicio.php
    |       vergraficapulso.php
    |       vergraficapulsopacientes.php
    |       vergraficatemplinea.php
    |       vergraficatemplineapacientes.php
    |
    +---medicos
    |       leerbasedatosecg.php
    |       listapaciente.php
    |       medidaspeticiones.php
    |       miinformacion.php
    |       paginainicio.php
    |       vergraficapulso.php
    |       vergraficatemplinea.php
```

```
|
\---pacientes
        informacionmimedico.php
        leerbasedatosecg.php
        miinformacion.php
        paginainicio.php
        vergraficapulso.php
        vergraficatemplinea.php
```